Towards an Intelligent Learning Environment for Melody Composition Through Formalisation of Narmour's *Implication-Realisation Model*

Thesis

UNRESTRICTED

Towards an Intelligent Learning Environment for Melody Composition Through Formalisation of Narmour's *Implication-Realisation Model*

Matt Smith

BA (Hons) Computing in Business, CNAA (Huddersfield Polytechnic), 1989
MSc Applied Artificial Intelligence, University of Aberdeen, 1990

Date of submission: 28 April 1995
Date of award: 28 February 1997

Submitted for the degree of Doctor of Philosophy
Department of Computing, Open University

1995

Abstract

This thesis describes research with the ultimate goal of finding ways to use artificial intelligence (AI) to encourage and facilitate melody composition by musical novices. The work described in this thesis forms the groundwork for the development of intelligent learning environments for novice composers in this domain.

There were two stages to the research. The first stage was the formalisation and testing of an existing analytical theory of melody. This theory offers an explanation of how musical listeners break-up a melody into "chunks", and hear some notes as more important than others. The theory enables analysis of melodies in a hierarchical fashion. The formalisation process involved the implementation of a parser to create hierarchical analyses, and comparison of published analyses based on the theory with those created by the parser from the same melodies. From these results a critical evaluation of the analytical theory, and the parser, is presented.

The second stage of the research involved extending the parser with constraint-based generation techniques. One result is an AI tool (called MOTIVE), which can generate melodies given an analysis (from the parser, or constructed by the user) and a set of constraints to be applied at each hierarchical level. The features of the tool are presented, and a general architecture for an intelligent learning environment is proposed, within which MOTIVE would reside, which shows how the formalised analytical theory from the first part of the work could be used educationally by novice composers of melody.

Keywords

Artificial Intelligence, Music, Composition, Melody, Constraints, Learning Tool.

Dedication

To the memory of Daran Coates

colleague, musical mentor, friend.

Acknowledgements

Simon, without whose enthusiasm, support and advice I would neither have been encouraged to begin this work, nor been able to keep going at the times it all seemed too much — thanks for everything.

John Cook — thanks for comments on early versions of this work, and the useful meetings we've had.

Thanks to the Edinburgh crowd: Geraint Wiggins, Alan Smaill and Eduardo Miranda — for early encouragement in this research and collaboration on the workshop and book.

Thanks to all my family (cats included!) — special apologies to Mark and Jill, this thing has stopped me spending more time writing to you (no excuse now...).

Special thanks to friends from the various parts of the country I've been around while doing this work, for support and generally making life fun along the way — especially Gillian, Martin and Claudia from Aberdeen, and Kate, Sas, Jon, Dave, Nora and the gang at Milton Keynes.

Thanks, congratulations and encouragement to all my fellow PhD'ers at both Aberdeen, Milton Keynes, Winchester and London.

Susan, for a continuing and rewarding friendship, and research support along the way. Looks like I just beat you.

Bam-Bam's mum — thanks for everything.

A special mention to family number two at Eastleigh — Helen, Rus and Jack, for providing welcome diversions and a home.

Dawnie, this is my albatross put to rest — your turn now.

Sinéad, thanks for doing the final proof read for me.

Thanks to all in Computing and Maths at King Alfred's, for welcoming me, and supporting me while I completed this work. Also to my students at KAC, for being guinea pigs for my lecturing — keep on juggling.

This research has been financially supported by SERC research studentship 90311096, and a grant from the Department of Computing at the Open University.

I would like to thank the following departments for use of equipment and facilities at various stages of this research: the Department of Computing Science, King's College, University of Aberdeen; the Department of Computing, Open University; the Computing and Mathematics Group, King Alfred's College, Winchester; and the School of Computing Science, Middlesex Unversity.

Finally thanks to all who have provided support and encouragement to me over the last few years whom I've neglected to mention by name.

Table of Contents

List of figures

Chapter 4: Formalisation of Narmour's Theory

Chapter 5: The parser

Chapter 6: Testing Narmour's theory

Chapter 7: MOTIVE: A tool for constraint-based generation of melodies

Chapter 8: Conclusions, critique and further work

List of appendices

Chapter 1:
Introduction

"If one is to consider the 'cognition of basic melodic structures' one must do so in terms of what we currently know about cognition. Artificial Intelligence and cognitive science have recently provided us with powerful intellectual tools. As the philosopher John Pollock has observed regarding the study of epistemology (Pollock 1989), the availability of these tools implies that we can no longer be content to speculate about cognition from the comfort of an armchair. We can build and experiment with concrete models, thus honoring Narmour's driving ambition to be scientific about this whole affair" (p. 47, Smoliar 1991)

1.1. The research problem

The ultimate goal of the research described in this thesis is to find ways of using artificial intelligence (AI) to encourage and facilitate melody composition by musical novices. In this thesis we describe work which provides the necessary groundwork for one approach to the development of such educational systems. Our ultimate research goal focuses on those students with a small amount of previous musical training, who are trying to teach themselves to compose[1]. As stated by Levitt (1985), the main goal of music composition is to "compose

[1] That is students who have a simple understanding of metre, harmony and common music notation, but with no experience of composition, and little or none of analysis.

something interesting". Thus there are no obvious restrictions of the (compositional) tasks that the novice composer may wish to attempt. Since the analytical musical theory central to this thesis is based on the Western Tonal Music (WTM) paradigm, the work in this thesis limits itself to such musics — although, where appropriate, applications of the research to other musical paradigms are discussed.

When novices attempt to construct a creative artifact, such as a melody, two problems they can encounter are:

- **problem 1: "Where do I begin?"**

  there are generally few attributes of the task that are given — i.e. the student is given a *carte blanche*,

- **problem 2: "When have I finished?"**

  the criteria for what defines successful task completion are poorly defined — probably in terms of a small number of explicit and implicit constraints.

A third problem may also be encountered by novice composers, after having begun and before completeing:

- **problem 3: "Where do I go from here?"**

  having completed a part of the task (for example, having constructed a component of the artefact, such as a melodic figure or phrase) the student is unsure how to go forward.

At the end of this thesis we shall argue that the work we have done provides a sound basis on which to develop an intelligent learning environment to offer ways for novices to begin to overcome all three of these problems.

The research work described in this thesis follows a particular methodology (described in section 2.4.3. of the next chapter). In brief, the methodology required the identification and computational instantiation of a musical theory, for the development of a computer tool around which an intelligent learning environment can be built. It was important that the musical theory chosen was not limited, for example to particular musical genres, and was one that has some psychological basis. A candidate musical theory was identified, and although this theory has been accepted as an important one by the musicological community (see detailed reviews in Chapter 3), the theory required formalisation before it could be computationally modelled. For

reasons we discuss in the following chapter (Chapter 2), the chosen musical theory was one for melodic analysis, not composition, so a framework had to be developed for using the computational instantiation of the theory as the basis of a computational tool for melody generation.

1.2. The approach adopted

In this thesis a clarification, extension and formalisation of an analytical theory of melody is described. The process has allowed the unambiguous representation of the theory, and the implementation of a computational parser based on the formalised theory.

The work described in this thesis embodies the cognitive science method of building a computational model of a cognitive theory in order to clarify and test the theory (Slack, 1984). We have then used the formalised theory to develop foundations for tools allowing users to benefit from the theory without having to master its technical details. This approach in AI and music research is summed up in Otto Laske's paper on what he calls "cognitive musicology":

> *"Briefly, computer programs serve two distinct functions in musicology. First, they serve to substantiate hypotheses regarding musical knowledge, and second, they are the medium for designing structured task environments (such as programs for interactive composition). ... What is more, systems, once built, can serve as a focused research environment for a deeper understanding of the activity concerned."*
>
> (p. 45, Laske, 1988)

Following the formalisation of the theory and the development of the parser, we describe in Chapters 7 and 8 how the pre-requisites have been solved to allow such a formalised theory to be used as the basis for intelligent learning environments (ILE) in the domain of melody composition (for novices). Chapter 7 describes constraint-based extensions to the parser. The reasoning behind the development of a tool for constraint-based generation is to take advantage of people's more developed critical skills (as argued by Perkins, 1981, for example); this approach is related to Johnson-Laird's (1988) multi-stage model of creativity. These issues of constraints and the development of tools and educational systems for creative activities are considered in Chapter 7, and in the further work section of Chapter 8.

1.2.1. The use of an analytical theory for generation

An interesting feature of our research is how an analytical theory of melody has been used as the basis for a tool for melody generation. It will be argued in Chapter 7 that the constraint-based generation approach used to develop our AI tool has implications for other problems, where methods of analysis are more understood than methods for generation. A general principle has been applied — that an analysis can always be used as a template for generation, and that this template should be able to serve as the basis for re-generation of a certain class of objects that are related to the original object analysed in certain specific ways. How effective the template is at generating objects of interest depends on many factors, and such factors are explored in Chapter 7. A description is presented in Chapter 7 of how the application of a number of different kinds of constraints at hierarchical levels can guide the generation of new melodies from an analysis of an existing melody, and how such constraint satisfaction can fit within an intelligent learning environment.

The parser described in Chapter 5 was developed with eventual constraint generation in mind. Many of the parser procedures are used unchanged for the constrained generation of melodies.

1.2.2. The chosen analytical theory of melody

The analytical theory of melody upon which much of our research is based is Eugene Narmour's *Implication-Realisation Model* (1990, 1992). The theory has been acknowledged by the musicology community as a significant addition to the tools available for melody analysis (for example Butler 1992, and Cumming 1992). The theory is hierarchical, with rules of inference applied recursively upwards in a hierarchical analysis generated from a note-by-note parsing of a melody in chronological order. The theory is defined in terms of declarative structures and rules of inference, and has been described to a fine degree of detail — making the theory very amenable for computational instantiation. One of the two rules of inference is based on principles of the Gestalt psychology of perception (e.g. Koffka 1922, 1935; Katz 1951), and the hierarchical nature of the analyses and independent modelling of the parameters of melody is based on Fodor's (1983) modular model of cognition. A discussion of Narmour's use of Gestalt psychology is included in the summary of the strengths and weaknesses of the *Implication-Realisation Model* in the next chapter. Narmour's theory is attractive because he attempts to base his model for music analysis on a foundation of the cognition of perception; in the words of Naiomi Cumming (1992, p. 355) "*Narmour makes music theory a sub-discipline of cognitive psychology*".

Like all theories, the *Implication-Realisation Model* does have a number of limitations. A full discussion of the limitations of the theory is presented in Chapter 3. The main limitations of Narmour's theory include: the informal description of the theory (as presented by Narmour); the way that the top-down aspects of the model are only outlined, and it is these aspects that would allow musicologists to see how Narmour's listener-perception based theory relates to current theories of analysis based on accepted notions of harmony and form; Narmour's arguments for the existence of biologically "hard-wired" perceptual scales, of which there is increasing evidence against (for example see the summary of criticisms from Smoliar, 1991).

There is preliminary support for the *Implication-Realisation Model* from empirical psychological experiments (for example Krumhansl 1991; Krumhansl & Shellenberg 1990; Shellenberg & Krumhansl 1991). Despite a number of criticisms and weaknesses in the theory (and more fully discussed in Chapters 3 and 4 of this thesis), Narmour's theory stands out as the most appropriate in many respects for research into the development for computational tools to aid novices in melody analysis and composition. The strengths of the theory are explored in the following two chapters (especially in sections 2.3.5 and 3.12). Briefly the main strengths of the theory for our research are as follows: it is based on a psychological model of perception; the theory analyses music in terms of intrinsically melodic concepts (unlike the two main alternative strong musical theories discussed in Chapter 2); the theory has been described in considerable detail by Narmour; the two central concepts for analyses (melodic contour and interval size) can be simply represented and musical novices appear to easily understand them; and the theory is applied with a chronological, note-by-note technique very amenable to being modelled as a computational parser.

1.2.3. The interdisciplinary nature of the research

The research in this thesis draws from work in four main disciplines: Artificial Intelligence, Music, Psychology, and Education. Figure 1.1 is an informal illustration of the interconnections between these disciplines. The figure loosely illustrates how MOTIVE (the AI tool for melody analysis and generation described in this thesis) relates these four disciplines.

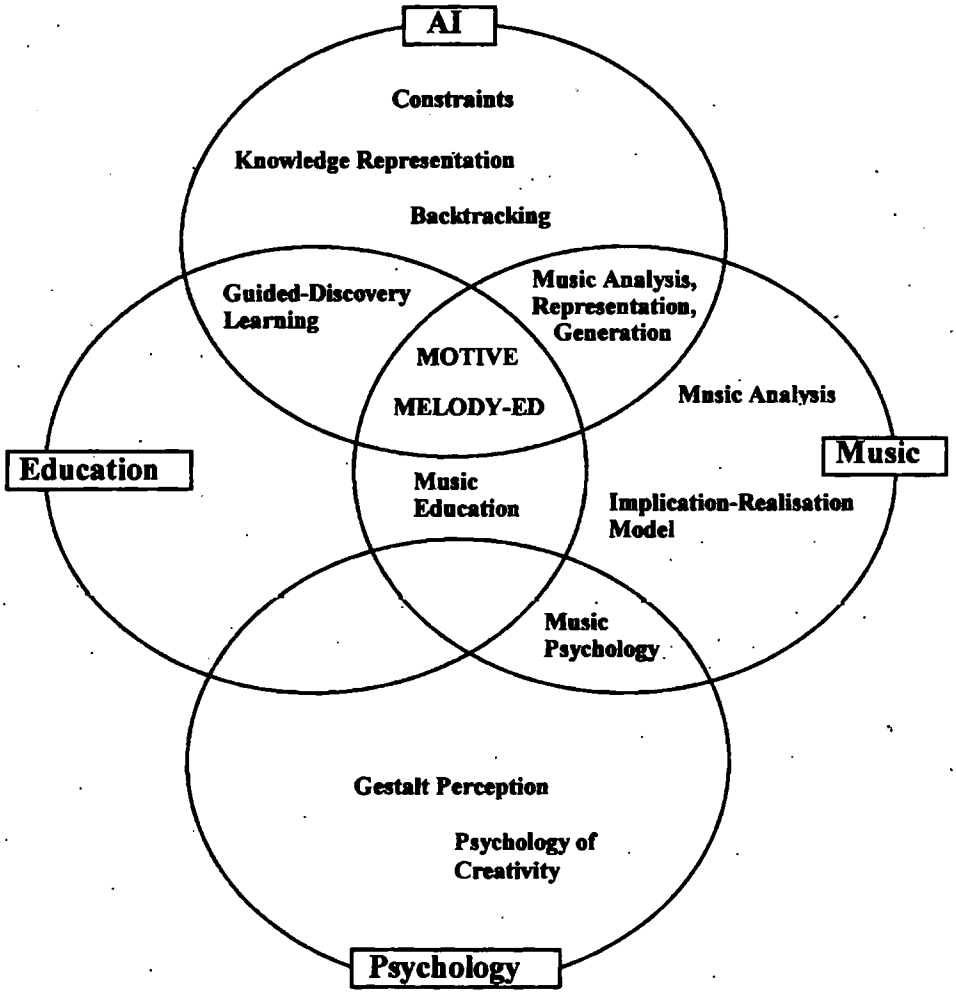

Figure 1.1: The interdisciplinary nature of this research.

One of the appealing features of Narmour's *Implication-Realisation Model* for educational purposes is the fact the two parameters of melody most important to the theory are melodic contour and interval size between pitches — both of these parameters can be represented visually in straightforward ways[2]. Whether a direct manipulation interface to MOTIVE could effectively exploit the modality of motion in a satisfactory and principled way is not so clear, but audio and visual modalities appear to fit very well for these two parameters. Such considerations are investigated in more detail in Chapter 8.

---

[2] Other important concepts for the *Implication-Realisation Model* can also be modelled graphically (for example hierarchical metre following the example of Levitt 1985, and harmony using the pitch spaces of Lerdahl 1988), however, what is important in our argument is that the central concepts of melody for the two rules of inference of Narmour's theory are able to be simply represented visually.

1.3. Aims of the research, and summary of thesis structure

Chapter 2 provides a review of important work in the research disciplines relevant to this research, and states how the work presented in this thesis fits into the interdisciplinary field of Artificial Intelligence (AI) and Music. Chapter 3 presents the melody analysis theory on which much of this work is based (i.e. Narmour's *Implication-Realisation Model*).

The structure of this thesis after Chapter 3 is perhaps best described in terms of the four-stage approach adopted to provide the groundwork for achieving the goal of finding ways to use artificial intelligence to encourage and facilitate melody composition by musical novices. The four stages are described in Chapters 4 to 7 respectively (each is described in more detail in this chapter in the sub-sections whose numbers appear in parentheses):

- ***(§1.3.1, Chapter 4)*** ***stage 1:*** the clarification, extension and formalisation of an analytical theory of melody, by the systematic identification of ambiguities, gaps and assumed knowledge in descriptions of the theory, by quantification of theory statements and definitions, and the representation of key points of the theory by the use of a formal notation,
- ***(§1.3.2, Chapter 5)*** ***stage 2:*** the design and implementation of a computational model of the formalised musical theory (in the form of a melody parser),
- ***(§1.3.3, Chapter 6)*** ***stage 3:*** testing of the formalised theory, by comparing parses produced by the computational parser against published, hand-produced analyses from the author of the analytical theory, and
- ***(§1.3.4, Chapter 7)*** ***stage 4:*** design of a constraint-based tool (MOTIVE) which extends the parser so that it can generate families of melodies from a set of musical constraints.

The final chapter (Chapter 8), presents conclusions, discusses the contributions, critically evaluates the research described in this thesis, and gives suggestions for further work. The further work section in Chapter 8 includes the presentation of an outline architecture for an intelligent learning environment for novice composers of melody based on our formalised version of the *Implication-Realisation Model*. Each of the stages listed above is described in more detail in the remainder of this section.

1.3.1. The clarification and formalisation of an analytical theory of melody (Chapter 4)

An analytical theory for melody is described in this thesis — the theory is based upon a model of perception[3], whereby listeners process perceptual events in a number of different cognitive modules, simultaneously and independently. In this thesis the part of the theory describing bottom-up parsing of melodic events is clarified, a process which has involved the modification and simplification of the theory in some ways, to allow its unambiguous representation. The modifications have included making consistent the hierarchical promotion of notes and showing how this still models the promotion of important notes to the same levels found in Narmour's examples. Other modifications and simplifications are the introduction of a numerical measure for combined "closure" (the determination of note groupings and promotions), and the introduction of a scale for the metric importance of note onset times.

The clarification and formalisation process had the following goals:

- to track down any gaps in the theory (i.e. circumstances the theory does not state how to analyse),
- to draw attention to ambiguities, describe the alternative interpretations, and justify any decisions we have made during the formalisation process,
- to systematically quantify Narmour's statements and theory definitions (where numerical or symbolic values are required for a parser decision to be made and Narmour's presentation of the theory is not given in a unambiguous, quantified manner),
- to check the theory for consistency,
- to use a predicate-calculus formalism, where appropriate, to make concrete imprecise verbal descriptions of the theory, and
- to identify knowledge that Narmour assumes an analyst to possess, i.e. both knowledge about aspects of music in general, and specific procedural knowledge about how to parse melodies in a note-by-note fashion to apply his theory.

---

[3] The psychological validity of the particular theory, and reasons for choosing it, are argued in the next chapter.

1.3.2. A computational parser (Chapter 5)

Chapter 5 describes the parser that has been developed as a computational instantiation of the formalised theory. The requirements of building a complete, explicit computational model of the theory have led to insights that were not apparent when using written notations for the clarification of the theory. For example the procedural aspects of applying Narmour's theory note-by-note, and simultaneously at multiple levels in a hierarchy, are not described by Narmour in any detail — such aspects of the clarification and formalisation of his theory were a major pre-requisite for the parser's design.

1.3.3. Evaluation of the clarified theory (Chapter 6)

The clarified and formalised theory has been tested within the framework of the assumptions made during the formalisation, involving a comparison of Narmour's hand-generated example analyses with those produced by the parser for the same melodies. This chapter examines the possible reasons for each mismatch, and also discusses ways in which the formalised theory can be further tested and what has been demonstrated by our testing.

1.3.4. MOTIVE: A constraint-based educational tool (Chapter 7)

Extensions to the parser described earlier this chapter have been developed, to allow novice composers to generate melodies through the principled application of constraints at levels in a hierarchy. The hierarchy used for the generation of melodies is in the same form as the hierarchical analyses that the parser produces (i.e. an analysis defined by the clarified version of Narmour's *Implication-Realisation Model)* — thus from the parsing of a single melody, constraint-generation allows a *family* of melodies to be generated, which all have the same hierarchical analysis (either constructed from scratch by the student, or based on an analysis of an existing melody). These constraint-based extensions, plus the parser itself, form an integrated tool for melody analysis and generation (called MOTIVE).

1.4. Relevance of this work for other domains

It will be argued that the inherent complexity and open-ended nature of music makes it a good choice of domain for AI and education researchers, since the difficulty of the problems which arise when developing AI models for music analysis and composition call for innovative solutions, that may then be applied to certain other domains. Problems in traditional Intelligent Tutoring System (ITS) domains such as arithmetic (Sleeman 1983, Young & O'Shea, 1981) or

physics (White, 1981) tend to have well defined criteria for solutions, although are able to be viewed as tasks requiring creative problem solving. Holland (1989) argues the point as follows:

> *"We can postulate that in any knowledge-rich domain with a sufficiently large search space, searching the space can be treated as an open-ended creative task. Any domain at all, even arithmetic, electronics, physics or programming may be considered creatively when a requirement arises to extend the domain or reconceptualise it. Indeed, some teachers and psychologists might argue that extension and reconceptualisation lie at the heart of effective learning in general. For these reasons, progress towards architectures for tutoring creative tasks has implications for the discipline [ITS] as a whole."* (Holland, 1989, p. 6)

Chapter 7 of this thesis demonstrates how constraint-satisfaction techniques can be used to simplify the task of generating solutions for open-ended, creative tasks, in particular the types of tasks with which a novice composer of melodies is faced. So for domains where there is a large search space of possible solutions, or the criteria for good solutions to a problem are not clear, a constraint-based approach to solving creative problems could be applied. In addition to describing the constraint-satisfaction techniques in general, Chapter 7 describes the application of these techniques to MOTIVE as a tool for melody generation in particular, and a proposal is made for an educational framework for solving creative melody composition tasks (MELODY-ED) within which MOTIVE is used (outlined in Chapter 8).

1.5. Timeliness

A number of interesting developments in AI and music research have been reported in recent years, including:

- Narmour (1990, 1992) an analytical theory for melody,
- Smith (1990) Prolog implementation of Levitt's (1985) proposals,
- Holland (1989, 1990, 1991) constraint-based tutor for novice composers,
- Holland (1989) and Coates (1994) AI applications of Longuet-Higgins (1962) two-dimensional pitch space encapsulating many harmonic relationships,
- Baker (1989a and 1989b) implementation of a grouping analysis tool based on Lerdahl and Jackendoff's (1983) analytical theory of music,
- Lerdahl (1988) proposing a set of "pitch spaces" for harmonic analysis, and

- Levitt (1985) proposals for constraint techniques for melody generation.

The work on MOTIVE has used and adapted much of the above research. The above theoretical work also coincides with two important technological developments: the design of MOTIVE takes advantage of the speed of current personal computers allowing real-time interaction between humans and computers running AI languages (such as Prolog) via audio-graphical interfaces. Until recently personal computers were too slow for AI languages to run in real time, thus making the task of implementing AI systems much larger, since once a design was complete it would require rewriting in a lower level language (such as C). The second timely technological development is the emergence of low cost, high quality electronic musical instruments, and the establishment of the MIDI (De Furia & Scacciaferro, 1988) and General MIDI communication standards (Loy, 1985); the availability of this technology means that work such as MOTIVE is open to widespread practical application. Reimer (1989, cited in Cook 1994, p. 20) describes computers and low cost keyboards as: *"... the birth of mass composition opportunities"*.

To the author's knowledge, the computational model (M-PARSER) is the first substantial computer model of Narmour's theory. A research project was begun, which initially was leading to a sub-symbolic model of aspects of Narmour's theory, however the project changed direction and no such model was ever developed (Griffith, 1994).

1.6. Roots of the research

The first stage of the research (the formalisation of Narmour's theory and subsequent development of the computational parser) was inspired by a single piece of work — Narmour (1989, 1990, 1992). A number of the additions to the parser were influenced by Levitt (1984, 1985) and Smith (1990).

The overall methodology of the research is closely related to that of Holland (1989). In addition Holland's constraint-based approach (1990, 1991) has been an important influence on our proposals for how the MOTIVE tool might be used educationally, and on our outline architecture for a constraint-based intelligent learning environment proposed in the further research section of Chapter 8.

Chapter 2:
Literature Review

> *"The task of musical composition is less well understood and less well researched than many other complex cognitive tasks... One reason for the comparative lack of research into musical cognition is that the underlying processes are assumed to be creative as well as analytical, and hence less penetrable using standard scientific methodologies."* (Colley et al. 1992, p. 124)

This chapter presents a critical review of research from a number of disciplines. The chapter, after presenting a review of Johnson-Laird's (1988) model of creativity, breaks down into the following distinct sections:

- melody composition (and the learning of such skills),
- the choice of an appropriate theory of melody for our research,
- previous work on AI and music education, and
- other related research.

Johnson-Laird's work is reviewed before the section on melody composition, since parts of that section make reference to his model of creativity.

This review presents a sequence of stages describing how our work was guided by existing research, and in some cases the lack of it. In addition to the research reviewed in this chapter, further reviews can be found in Chapters 3 and 7. Chapter 3, in presenting a description of Narmour's *Implication-Realisation Model,* also includes a review of critiques of Narmour's

theory. In Chapter 7 a review of AI techniques for generate-and-test and constraint-satisfaction is given.

The groundwork research described in this thesis needed to be appropriate for use in achieving our long term research goal (of developing an intelligent learning environment for novice composers of melody). For this reason we have included in this chapter a number of reviews of areas such as music education and the cognitive psychology of creativity — such work is cited in later parts of this thesis where the work we have done is evaluated in terms of our final goal.

2.1. Johnson-Laird's model of creativity

Johnson-Laird (1988) proposes a view of creativity with the following three characteristics (the first two of which are important influences of the work in this thesis):

> *"First, like all mental processes, it [*creativity*] starts from some given building blocks.*
>
> *Second, the process has no precise goal, but only some pre-existing constraints or criteria that it must meet.*
>
> *Third, a creative process yields an outcome that is novel for the individual, not merely remembered or perceived, and not constructed by rote or by a simple deterministic procedure."* (Johnson-Laird 1988, p. 255)[1]

For music, the building blocks to which Johnson-Laird refers (point one above) could be the "primitives" of Western Tonal Music, such as scales, standard cadence chord sequences, and metre. However, even given these musical building blocks the number of possible compositions is extremely large, and the task of composition still has no starting point — i.e. a student is still initially given a blank score. The difficulty of music composition tasks due to starting with a blank score is an example of the first type of problem faced by novice composers described at the beginning of Chapter 1 (i.e. "where do I begin?").

---

[1] "*[*creativity*]*" has been added for clarity.

The second of Johnson-Laird's points is that creative processes such as music composition have few objective criteria for what is to be done — Levitt refers to the goal of music composition as simply "*compose something interesting*" (Levitt, 1985). This means that while performing the process of composition, a student is unable to compare the current state of the composition with a well defined task specification, to test whether the task is complete. The fact that novice composers have trouble deciding when they should stop working on a melody is an example of the second type of problem faced by novice composers described at the beginning of Chapter 1 (i.e. "when have I finished?").

If an interactive learning environment for the construction of creative artifacts is to be based upon the view of creativity suggested by Johnson-Laird, it should be designed around the main features of his theory, therefore such a system should:

- assist in making the building blocks for the task explicit, and readily available to students,
- assist in making explicit the criteria for task completion, and provide students with the means to evaluate artifacts against the recognised criteria.

2.1.1. Multi-stage creativity

Johnson-Laird goes on to critically discuss a number of computational architectures for creative processes, proposing a "multi-stage" model as explanation of why humans are often much better critics of creative artifacts than generators. Johnson-Laird (referring to Perkins, 1981) suggests that although constraints are used in both the generation and criticism of creative artifacts, it may be that the constraints used for criticism are not available for use by the generative mechanism — otherwise the critical constraints could be included to make the generative mechanism much more efficient. Informal observations suggest that novice composers are much more able to criticise a set of melodies (or melody fragments) in terms of structure, harmony and metre, than to generate such melodies.

Johnson-Laird says of multi-stage creativity and its relationship to constraints the following:

> *"Multi-stage creativity uses constraints both generatively and selectively. ... the paradox of creativity is that people are better critics than creators: their knowledge is more readily available to them for judgment than for generation. Many creative achievements thus depend on a multi-stage process in which an initial idea is*

> *generated and then goes through a progressive series of revisions guided by constraints."* (Johnson-Laird 1988, p. 258)

Since humans seem to work creatively in a multi-stage way, the design of an interactive learning environment for creative tasks could be more effective if it helps make explicit, and separate, those constraints used for generation, and those for selection. In addition, such an architecture should aim to allow students to move easily between generative and selective activities.

2.2. Melody composition (and the learning of such skills)

> *"The reader should be warned that composition is the least studied and less well understood of all musical processes, and that there is no substantial psychological literature to review"* (Sloboda 1985, p. 103)

At the outset of the research described in this thesis a search was made for existing work on the cognitive processes involved in composition, and how people learn composition skills. However, there appears to be very little formal literature on either the process of melody composition, or how students wishing to develop such composition skills learn, and may be assisted in such learning. Whilst much has been published on composition, in general, e.g. Schoenberg (1943, 1967) or Wuorinen (1979), and melody in particular, e.g. Warburton (1960), very little has any theoretical basis in either the cognitive psychology of music, or the cognitive psychology of learning. This section breaks down into reviews of research in the following areas:

- the cognitive process of composition (and how it may be studied),
- expert / novice comparisons, and
- studies of children's compositional strategies.

2.2.1. The cognitive process of composition

Although there is much literature about compositions (i.e. the product of composers), little research has been conducted into studying the cognitive **process** of composition itself. Sloboda (1985, in his book on the cognitive psychology of music in general), suggests four methods of inquiry into the cognitive psychology of composition (p. 102):

- examination of the history of a particular composition (as displayed in the composer's written manuscripts),
- examination of what composers say about their own compositional processes,
- 'live' observation of composers during a session of composition, and
- observation and description of improvisatory performance.

In examining evidence from studies using these four methods Sloboda begins to outline the processes involved in composition (as extracted from the writings of composers). He is quick to point out, however, that his diagrams and explanations are not a theory of the compositional process. Such computer composition programs as Sundberg and Lindblom's (1976) are criticised, since they do not attempt to model any approximation of the cognitive processes involved in human composition — especially in that such programs have no "verification" phase, where part or full compositions are tested against criteria, for possible rejection or modification (i.e. the programs produce a solution that is "right first time").

2.2.2. Expert / novice comparisons

Colley et al. (1992) report a comparative study between expert and novice composers. Although the task was harmonisation (rather than melody composition), the cognitive task of composition in general is discussed, and it would seem likely that the expert-novice comparisons are applicable to other composition tasks. As stated in the report, a major problem (and perhaps another reason for the lack of research into musical composition) is that the end product is difficult to evaluate, since there are no explicit goals against which to evaluate a composition.

In summarising the work of Reitman (1965), who concentrates on a constraint-based view of composition, and Simon (1973), concentrating on problem solving and reduction of the problem search space, Colley et al. state that at the beginning composition is an ill-structured, problem solving task, in which some initial constraints may be present (e.g. rules of tonal structure for a

genre). Other constraints are then identified and applied by the composer, and others emerge (i.e. earlier decisions constrain later decisions if the composition is to be structured and consistent). Both Narmour (1990, 1992, and summarised in Butler 1992, p.244) and Holland (1989, p. 144) propose frameworks of composition where the influence of existing music (from within the same piece, or style, or tonal paradigm) occurs simultaneously. The three frameworks mentioned (Colley et al., Holland and Narmour) are summarised in Figure 2.1 below. The issues raised by these frameworks are considered in more detail later in the thesis (Narmour's framework in Chapter 3, and constraints for composition in general in Chapter 7).

Colley et al. (constraints)	Holland (levels of knowledge)	Narmour (expectations & implications)
		stylistic primitives of musicalparameters
		tonal idiom
	musical plan level	
rules for tonal structure for a musical genre	style level	extra opus style (school)
constraints identified and applied by composer	style level	extra opus style (composer)
constraints identified and applied by composer	song level	intra opus style
constraints from earlier decisions in a piece	song level	intra opus style

Figure 2.1: Summary of influence of existing music on composition (and cognition) process.

A tentative hypothesis arising from Figure 2.1 is that the chronological order of influence (constraints) from existing music would be from top to bottom in the diagram — i.e. that before starting a composition a composer is aware (or unconsciously able to apply) constraints relating to tonal primitives and tonal idiom. As the composition is begun decisions are made and the composer chooses to apply the stylistic constraints of a musical genre, and initially

makes some compositional decisions such as themes, and the metric and harmonic framework for the piece. Once the composition is under way, previous decisions made in the piece constrain the composer if the piece is to have large scale structure and consistency. A weakness of this chronological view may be the constraints that relate to a particular composer, since some of these may well be (unconsciously) applied at all times by the composer. However, in the case of a composer identifying a constraint within a composition, and then satisfying it consciously from that point onward, the composer is exhibiting what Reitman (cited by Sloboda, 1985, p. 124) describes as "... *after-the-fact adoption of a convention which is consistent with what has already been done*", and this behaviour would be in keeping with the suggested chronological order of Figure 2.1.

Colley et al. draw two conclusions from Reitman's and Simon's work:

(1) the composer requires a knowledge base containing a substantial amount of domain-related information in order to be able to supply constraints for the task, and

(2) composition skill requires the identification and application of constraints.

These two conclusions could be considered instantiations, for the process of composition, of the first two properties of Johnson-Laird's (1988) model of creative process (reviewed later in this chapter), i.e.:

(1) like all mental processes, a creative process starts from some given building blocks, and

(2) the creative process has no precise goal, but only some pre-existing constraints.

The conclusions of Colley's et al. comparative study[2] can be summarised in the following four points:

(a) the expert demonstrated greater knowledge of features of the musical genre than the novices,

2 The task was the completion of a harmonisation of a four phrase Bach chorale (an adaptation of Bach's, "May what my God will come to pass", first stanza, Part 1, St. Matthew Passion (and Cantata 1)). The subjects were three novice composers, and one expert.

(b) the novices used (and frequently referred to) written prompts (such as the notes of a chord) while performing harmonisation,

(c) the expert took a broad overview of the task (while novices generally worked chord-by-chord, and had a preoccupation with technical detail), and

(d) the expert was able to apply the rules of harmony while simultaneously considering the shape of the composition.

From the point of view of constraints, and Johnson-Laird's model of creativity, the four points can be re-expressed as follows:

(a) the expert had more building blocks for the task (e.g. more knowledge of harmonisation techniques, a larger repertoire of past successful harmonisations to draw from, etc.),

(b) the novices had not automated some of the processes for applying constraints and building blocks to a task,

(c) the expert considered large scale, structural constraints of the task (the novices did not), and

(d) the expert was able to consider satisfying multiple constraints simultaneously (the novices were not able to do this).

Kratus (1991) summarises a study of ten students by Davidson and Welsh (1988), five with two years of college conservatory training, and five in their first year of training. The task given to the students was the composition of a modulating melody within half an hour, which returns to the original key (the melody was to modulate Cmaj — F#maj — Cmaj). The study results were very similar to Colley et al. above, the two main conclusions being:

- experienced students made local decisions about the melody while considering the piece as a whole, and
- beginners tended to work note-by-note, hardly considering the overall shape of the piece.

In passing, Kratus (1991) also briefly mentions a study by Bamberger (1977) — the study (a comparison of two college-aged students) concluded that *"... the student's understanding of musical syntax was reflected in their approach to composition."* (Kratus 1991, p. 96).

2.2.3. Studies of children's composition strategies

It would seem likely that research into the cognitive processes of composition (and the development of such skills) can benefit from studies of the progress and strategies used by children of different ages. In this section the findings of two studies of children's composition are summarised.

Kratus reports two studies of the process of composition by children. The first study (Kratus, 1985), involving groups of students aged 7, 9 and 11 years, examined the relative amount of time children spent on parts of the composition process, during a ten minute composing task. Kratus summarises his findings of this study as follows:

> *"Results indicated that 7-year-olds differ from 9- and 11-year-olds, in that 7-year-olds spend significantly more time exploring new materials as they compose and spend significantly less time developing and repeating their ideas. In addition, subjects who were able to play their songs the same way twice used significantly more repetition and less exploration than did subjects who could not replicate their songs."* (Kratus 1991, p.96)

The second study by Kratus (1991) aimed to analyse the different compositional strategies employed by children. The subjects were sixty children, again aged 7, 9 and 11. The subjects had no musical training, and after being introduced to a simple electronic keyboard were given ten minutes and asked to *"... make up a new song using the white keys of the keyboard."* (Kratus 1991, p. 96). Two school teachers were used as independent judges, to choose the ten most successful and ten least successful melodies. Songs were judged according to how closely their song could be repeated twice by each student, and consideration of how each song formed a cohesive whole, and the use of interesting melodic and rhythmic patterns. The compositions of students were analysed in terms of the amount of time students spent using different compositional strategies (strategies such as: using adjacent scale steps, using larger intervals, changing either a pattern's pitches and rhythm, transposition of a pattern, repetition of pitches and patterns, repeating the complete song, spoken questions or statements by students, and periods of silence where the student neither plays nor speaks). The results can be summarised as follows:

- successful students developed patterns by extension and changing the rhythm (while less successful students, if they did develop, mainly used transposition),

- successful students used stepping and skipping initially, but infrequently later in the composition (less successful students tended to continue exploring throughout the composition period),
- repetition (of pitches and patterns) was more common in the successful students,
- every successful student demonstrated closure of the melody within the ten minute period (demonstrated by repetition of the whole melody) — in many cases closure was demonstrated within the first two minutes of the period, and
- speaking and silence occurred less frequently with the more successful students.

As Kratus points out, the results lead to some interesting conclusions:

- successful songs are the product of certain compositional strategies,
- such strategies are different to those used by less successful students,
- i.e. *"... the success of the product would appear to be dependent upon the nature of the process" (p. 102).*

Kratus suggests that research intending to improve the educational process for creative domains may well benefit by focusing more on the creative process than on the products of the process. He also asks the important question: *"Can the strategies used successfully by high-success students be taught to low-success students?"*. Although the study was exploratory, it suggests that the design of any educational system for music composition should include consideration of how to encourage student's use of successful compositional strategies. Whether adults, in the process of attempting larger scale compositional tasks, also use common strategies that lead to more successful compositions is an area for further study.

Although there have been a number of exploratory studies of the process of the cognition of the task of composition (as reviewed in this section), no formal theory for composition has been proposed. Neither has any theory for how people learn the skills of composition been put forward. The main conclusions that can be drawn from the research to date is that composition appears to involve a process of constraint formation and satisfaction, that successful composers are able to satisfy multiple constraints simultaneously, and that successful compositions are those that have overall structure, as well as local adherence to constraints for a particular style.

2.3. Choosing an appropriate theory of melody

Since no obvious choice of a theory of music composition (or music composition education) is available, as reviewed above, this section reviews a number of music analysis (especially melody) theories. Also reviewed are related computational models of such music analysis theories. The theories considered are:

- Grammars of melody in general,
- Schenkerian analysis (Schenker 1979),
- Lerdahl and Jackendoff's (1983) *Generative Theory of Tonal Music*, and
- Narmour's (1990, 1992) *Implication-Realisation Model.*

Before reviewing the above theories, it is useful to consider what criteria a music analysis theory needs to fulfill, to be useful for the kind of educational tool which is the aim of the research described in this thesis.

2.3.1. Criteria for theories useful to this research

There are two important influences on the choice of a theory of music analysis for this research — first, the theory is to be the basis of an *educational* tool; and second, the *analysis* theory is to be used for an AI tool for melody *generation* (i.e. composition)[3].

A theory for an educational tool

For the educational approach adopted in our research (and described in detail in Chapter 8) it is important for the theoretical basis of the melody composition and analysis tool to be understandable to the user — since the aim of the tool and educational environment is not just for users to be able to generate new melodies, but to develop as composers through increased understanding of melody analysis and the process of composition. This immediately discounts such sub-symbolic research as Bharucha (1994), Kaipainen et al. (1995), Leman (1989 and

[3] In the last ten or so years there has been move away from "expert" based intelligent tutoring systems (ITSs) as the only model for computer-based learning environments. The work in this thesis provides important groundwork for our proposed framework for intelligent learning environments. A discussion as to why an approach has been adopted using theories that can generate melodies is presented as part of Chapter 8.

1992), and Todd (1989), since the knowledge encoded in the networks is not in a form understandable to students or researchers.

A theory for a tool for melody generation

The task of developing a tool for melody generation from a theory for melody analysis has been introduced in the previous chapter (and Chapter 7 discusses the use of constraints for generation in detail). A theory from which a constraint-based generator can be developed needs to be expressible in terms of declarative constraints (such as grammars and theories expressed as rules).

2.3.2. Melody grammars

A number of grammars for melody have been proposed (for example Baroni & Jacoboni, 1978). The use of grammars for analysis and generation is well researched (for example in natural language systems, e.g. McDonald 1983, and Brady & Berwick 1983), and certainly grammars appear to be a promising candidate for use in the generation of melodies. It can be hard to draw the line between melody grammars and related formal approaches. In principle, there is nothing preventing melody grammars from being used in the type of research presented in this thesis, but in practice most systems put forward as melody grammars have been rather genre specific (for example: Baroni and Jacoboni 1978, Ulrich 1977, and Steedman 1983). The goal of our research is to design and develop an educational system for novice composers of melodies for Western Tonal Music in general.

2.3.3. Schenkerian analysis

Heinrich Schenker developed a hierarchical music analysis theory at the beginning of the twentieth century (Schenker 1979, translated from the 1935 edition). The theory was developed before Chomsky (1957) formalised his theory of grammars, but has a number of features in common with grammars developed for natural language processing based on Chomsky's research. Sloboda (1985, cited in Rowe's book 1993, p. 101) notes the similarity between the two theories:

> *"... their theories have some striking similarities. They both argue, for their own subject matter, that human behaviour **must** be supported by the ability to form abstract underlying representations."* (Sloboda 1985, p. 11)

Schenker's theory is strongly reductionist, i.e. a given piece of music is reduced to a single form, called the *Ursatz* (from a small set of possible *Ursatz* forms). Notable features of Schenkerian analysis are the strict hierarchical levels, and the recursive application of the theory (Rowe 1993, p. 100). The theory has a number of weaknesses:

- rhythm is not explicitly accounted for,
- the theory is not formally defined and has many aspects open to interpretation, (in fact Schenker believed that music analysis is a creative act in itself[4]),
- As pointed out by Narmour (1983, p. 182), *"Schenkerian theory thus derives the function of the pitches and the levels on which they appear by asserting* a priori *fundamental lines and harmonies and then seeing* a posteriori *how the various voices elaborate the foreordained theoretical premises"*, i.e. the Schenkerian process forces the music into an analysis based on certain stylistic assumptions (which may not be true for the style in which the piece was composed).

Schenkerian analysis was used as a starting point for the CHORAL harmonisation system (Ebcioglu, 1992). Ebcioglu's system uses a bottom-up parsing technique for the bass and descant lines of the choral being generated. He admits that his Schenkerian analysis knowledge base is unable to generate reasonable parses based on Schenker's theory, and the development of a computational model of Schenkerian analysis of a piece is a *"difficult basic research project in music analysis"* (p. 312). Smoliar (1980) also describes a computer program to aid Schenkerian analysts.

In some cultures and some historical periods melody can be considered more or less independently from harmony, and certainly for our research an important criterion for a theory to be used must be that the theory has an explicit description of the role of melody *per se* in music. The lack of importance Schenkerian analysis gives melody ruled it out as the theory on which to base MOTIVE, although as an educational aid to harmonisation Ebcioglu's CHORAL system may have potential, if the analysis knowledge base could be made more sophisticated and represent the harmonisation knowledge in a form suitable for communication to a student.

---

[4] It is not uncommon to find two Schenkerian analysts having different opinions as the analysis of the same piece of music (e.g. Narmour 1984, p. 185 — where a Schenkerian analyst (Lester 1979) disagrees with Schenker's own Schenkerian analysis!).

2.3.4. Lerdahl and Jackendoff's generative theory

The work of Chomsky and Schenker discussed above are the two main influences of Lerdahl and Jackendoff's (1983) *Generative Theory of Tonal Music*. The theory aims to generate a structural description of a tonal piece of music, as is formed by an experienced listener. The theory is based on four structural components, and takes the form of well-formedness rules (for generating possible structural interpretations) and preference rules (for choosing which to apply).

As with Schenker's theory, Lerdahl and Jackendoff's theory is hierarchical and the rules are applied recursively, reducing the piece to higher level structures as one moves up the hierarchy. Although there has been some psychological support for the theory (for example Krumhansl's experiments for predictions of phrase boundaries, 1983), the theory has also come under criticism (for example Peel and Slawson, 1984; Clarke, 1986; and Rosner, 1984). One major criticism of their theory is that the analyses resulting from the theory only attempt to represent the *final* state of a listeners cognitive representations — they make no attempt to model the real-time, changing representations a listener constructs and modifies. Clarke highlights another problem, about the lack of support for the higher parts of their reduction trees:

> *"Evidence for the highest level in this structure is rather sparse, and is confined to statements by a number of composers (Mozart, Beethoven, Hindemith) which indicate that they were able to hear (or imagine) their own compositions in a single 'glance'"*
>
> (Clarke 1988, pp. 2-3, cited by Rowe 1993)

Lerdahl and Jackendoff's *Generative Theory of Tonal Music* is the basis for the AGA ("Automated Grouping Analysis") system (Baker 1989b). Baker uses a chart parser technique to extract grouping structures of piece, exploiting the *"relationship between grouping structure, parallelism and time-span reductions"* (Baker 1989b, p. 313). Baker uses the AGA system for the purpose of identifying grouping structure of a melody, which can then be used as the basis for an educational dialogue with a student about how to interpret the piece for performance (Baker 1992).

Perhaps the largest problem with the idea of using Lerdahl and Jackendoff's generative theory as the basis for a tool for melody composition is the lack of importance the theory gives to melody. Rosner (quoted in Narmour 1990, p. 4) says of Lerdahl and Jackendoff's theory:

> *"melody is almost a wraith in this theory."* (Rosner 1984, p. 285)

2.3.5. Narmour's Implication-Realisation Model

Narmour's (1990, 1992) *Implication-Realisation Model* is described and criticised in detail over the next two chapters. However, it is useful to summarise why this theory stands out from those above.

Narmour's theory is specifically about melody, and how listeners process the notes of a melody. His theory focuses mainly on two aspects of how listeners process melodies: first, how features of the notes of a melody and the harmonic and metric framework influence the grouping of notes by a listener; second, under what circumstances some notes are treated as more structurally important than others. The *Implication-Realisation Model* is based on the interaction between two systems of cognition: bottom-up, perceptual implications, and top-down stylistic influence. Narmour's book (1990) describes in detail the perceptual hypotheses on which the bottom-up cognitive system is based, although his descriptions are not always complete nor unambiguous, as discussed in Chapter 4.

The attractiveness of Narmour's theory stems partly from his detailed descriptions, and partly from the linguistic nature of the theory. Both mean that the theory lends itself to computational instantiation.

The theory is attractive also, because the melodic grouping structures are described in terms of simple concepts (such as interval size, registral direction, and similarity and differentiation of such values). These concepts are easy to grasp — it will be argued in Chapter 8 that the simplicity of the parameters, and students' existing experience with concepts of similarity and differentiation, make MOTIVE (our AI tool based upon Narmour's theory) educationally attractive. These simple, fundamental concepts to the theory probably make the learning and application of the theory more straightforward than the other theories described in this chapter. Although concepts (such as metre and harmony) do complicate the theory, the knowledge required for these aspects of the *Implication-Realisation Model* is similar to such knowledge required for theories such as Schenkerian analysis and Lerdahl and Jackendoff's *Generative*

Theory of Tonal Music. However, as discussed in Chapters 4 and 8, the simple, fundamental concepts of the *Implication-Realisation Model* provide a framework within which more sophisticated musical concepts can be introduced.

The theory has a number of other strengths. For example, by virtue of modelling musical perception in terms of intrinsically melodic concepts the *Implication-Realisation Model* is potentially more general than either of the other two major theories (Schenkerian analysis, and Lerdahl and Jackendoff's generative theory of tonal music). This generality stems from the fact that as well as being applicable to music from a tonal harmonic culture, the theory can also be applied to other musics, for example music from periods before the establishment of tonal harmony, and perhaps music heard by listeners who are not used to hearing tonal music. In this thesis we describe the design and development of a constraint-based tool for the analysis and generation of melodies — we argue that such tools based on melody-based theories, rather than harmony-based theories are more likely to aid users and learners in the understanding and manipulation of melodies.

One of the fundamental rules of inference of the *Implication-Realisation Model* (both of which are described in the next chapter) is based on principles of Gestalt psychology. There appears to be some support for such a basis for a cognitive theory of music analysis. Although some traditionalists claim Gestalt laws are inherently top-down (e.g. Bower & Hildgard 1981), others disagree and argue that Gestaltism contains a mixture of top-down and bottom-up aspects (e.g. Pomerantz 1981, p. 163, cited by Narmour, p. 63). Criticisms of Gestaltism (Marr 1982, p. 196, cited by Narmour 1990, p. 69) highlight the way the theory maintains that primitives are meaningless, then goes on to include inference rules based on the combination of primitives (such as the law of similarity). Marr talks of how Gestaltists failed to "*appreciate the complexity of functions that can be computed by local interactions*" (Marr, p. 196). Narmour cites Deutsch (1982a, 1982b, cited by Narmour 1990, p. 63) as additional defense of the relevance of Gestalt laws for the study of the cognitive psychology of music. Further support for the use of Gestalt psychology for the rules of inference of the *Implication-Realisation Model* is suggested by Butler (summarised in section 3.10.3 of the next chapter). Narmour attempts to follow the work suggested by Meyer (1956) in applying bottom-up aspects of Gestaltism in the formulation of a cognitive theory of melody analysis. In fact the bottom-up analyses of the *Implication-Realisation Model* could be seen as an attempt to take such an approach to an extreme, whereby the entire analysis is based on Gestalt concepts such as

similarity and proximity.

In the same way that Cumming (1992, p. 355) says Narmour treats music theory as a sub-discipline of cognitive psychology, we would argue that Narmour makes harmony, in so far as it relates to his theory, a sub-discipline of melodic closure. Narmour separates such top-down structural concepts as harmony, metre and style from the bottom-up cognitive processing by a listener. Narmour's model (described as a *duplex* model by Butler, 1992) models how learned concepts of harmony, metre and style can structure the grouping of music — such top-down structuring works simultaneously, and perhaps conflicts with, hard-wired, low-level, bottom-up perceptual processing.

This concludes our summary of music theories and description of the reasoning for the choice of Narmour's *Implication-Realisation Model*.

2.4. Previous work on AI and music education

A useful way to view research on AI and music education is from two perspectives:

(1) *intelligent tools and environments to aid composition* — environments and tools to aid students in some musical activity (i.e. cognitive support systems (Sharples & O'Malley, 1987) for music), whose focus is the provision of appropriate tools for the task rather than prescriptive or dialogue-based tutorial interaction, and

(2) *systems guiding the student, based on pedagogical principles* — pedagogical systems, aiming via some educational "agent" to teach, tutor, coach or act in some co-operative fashion to guide students' use of tools to perform some task.

The two types of system functions described above will be described by the terms: (1) musical cognitive support systems; (2) pedagogical musical systems. A number of recent research projects are reviewed below, from these two perspectives.

2.4.1. Musical cognitive support systems

Two recent environments for composition and performance are *Harmony Space* (Holland 1989 and 1994), and *Slappability* (Oppenheim 1994). The design of these environments were both motivated by attempting to make interactive composition and performance more accessible by

providing users with powerful metaphors for the human- (musician-) computer interface. Holland's *Harmony Space* was developed from two pieces of work (Longuet-Higgins 1962 and Balzano 1982) that both result in a two-dimensional spatial representation of pitches, which when implemented as a direct manipulation tool allow many harmonic relationships to be easily understood and used during real-time harmonic activities (such as composition and accompaniment). Oppenheim's *Slappability* came from an attempt to make an existing system of software for composition and performance (DMIX, Oppenheim 1993) more accessible to non-technical users, and also to provide a common metaphor for a diversity of different musical representations and activities. The *Slappability* system successfully combines numerous different representations of a piece of music (as digitised audio, as symbolic intervals and contours, as an algorithm etc.) with a small number of polymorphic operations to combine different representations — for example a simple algorithm can be "slapped" onto a symbolic, interval-based representation of a melody to produce a new interval-based melody with the original intervals changed by the algorithm.

2.4.2. Pedagogical musical systems

A number of educational systems for composition have been proposed and developed — including Thomas' (1985) *Vivace*, Holland's *Harmony Space* (as mentioned above), Baker's (1989c) KANT/GRAF system applied to dialogues about musical structure, and the COLERIDGE system (Cook 1994, Cook & Morgan 1995) for melody composition. The trend illustrated by this research is from restricted, rule-based systems (such as *Vivace*), to systems that either present musical concepts in intuitive ways via the interface (such as *Harmony Space*), or systems concentrating in promoting higher-order thinking skills (Cook's and Baker's work).

2.4.3. Holland's environment and methodology

Another aspect of Holland's (1989, 1991) research is the design (and partial implementation) of a constraint-based educational environment for music composition. The nature of constraints and the compositional process have been outlined earlier in this chapter, and the identification of such constraints (and the earlier work by Holland), has driven much of the design of MELODY-ED, the framework for intelligent educational environments described in Chapter 8 of this thesis. The following sub-sections summarise Holland's work, describing both his methodology and his MC intelligent learning environment.

Holland's Methodology for micro-world development

The methodology described by Holland (1989), and to some extent adopted for our research, can be summarised as taking the form of the following stages:

(1) the identification of an appropriate music theory (or theories) upon which a cognitive support tool could be based,

(2) the formalisation of the musical theory in a form appropriate for instantiation as a computational model,

(3) the development of a computational model of the formalised theory as a tool for the analysis and generation of pieces or fragments of music,

(4) investigation of the educational benefits experienced by novice composers (both in terms of understanding of musical theory and increased sophistication in compositions produced).

We have essentially followed the first three of the steps, and then, since the computational parser itself is insufficient for use as an educational tool for melody composition, we have moved onto the design of an intelligent learning environment for melody composition, and show how the computer model of Narmour's theory can be used as part of a constraint-based tool for melody generation.

Holland's tutor for music composition

Holland (1989, and Holland & Elsom-Cook 1990) describes a system (which he calls "MC") for music composition, comprising three microworlds and a knowledge-based tutor. One of the microworlds is the *Harmony Space* tool, resulting from the methodology discussed in the preceding section. Holland's MC framework is one example of the *Guided Discovery Tutoring* framework proposed by Elsom-Cook (1989). The general architecture for MC is reproduced in Figure 2.2 (Holland's Figure 1, Chapter 8, p. 145, 1989).

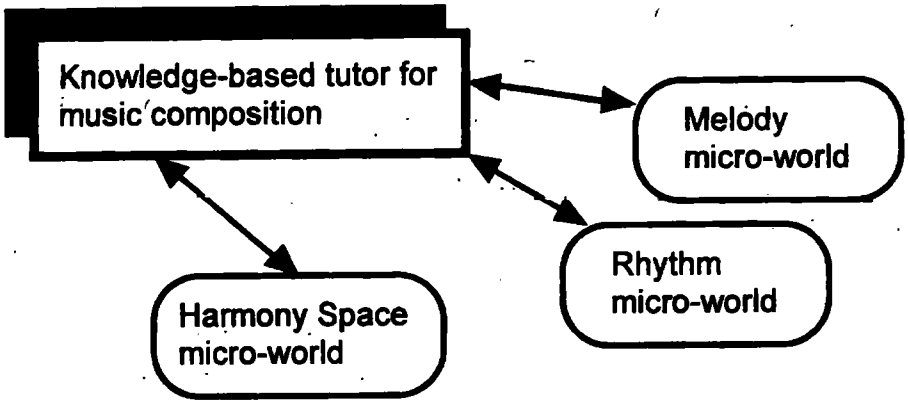

Figure 2.2: General architecture of Holland's MC framework.

The MC tutoring system embodies a constraint-generation approach to music composition, illustrating a number of interesting features:

- description of approximations to existing pieces of music in the form of constraints (called "chunks" in MC) (the constraints being variations on a set of default generators),
- outlining of how the manipulation of such chunks can be used to implement a "Paul Simon" method of taking an existing piece of music and making sensible modifications to create new pieces of music,
- description of how chord sequences can be expressed in terms of musical plans (which can be implemented, for example, as paths in Holland's *Harmony Space*).

The MC architecture illustrates how the manipulation of a small number of constraints can result in plausible chord sequences and compositions based on the chord sequences. Both the use of a set of plausible, *default constraints* for the generation of music, and the potential of the "Paul Simon" method for novice composers have been influences of the design of our MOTIVE tool.

2.4.4. Summary of AI and music education research

Ideally AI systems for music education combine the functions of being both effective cognitive support systems, and providing a means for the development of musical knowledge and skills. In some cases these two functions are performed by the same artifact, such as Holland's (1989) *Harmony Space*, which as well as being an effective tool for harmonic composition and analysis, uses a representation system that encourages the learning of many fundamental

harmonic relationships in tonal music.

The MOTIVE tool described later in this thesis aims to be both an effective cognitive support system and be an educational tool, in that it encourages a constraint-based view of the composition process, and is a means of using an formalised version of Narmour's *Implication-Realisation Model* of melody.

2.5. Other related research

In this section two pieces of research are summarised. These are:

- Levitt's (1984, 1985) hypothetical constraint language for musical dialects, and
- Lerdahl's (1988) model of Pitch Spaces of increasing consonance.

Each of these pieces of work has influenced the work described in this thesis. Levitt's work formed the basis of many musical concepts being computationally modelled as declarative constraints. Lerdahl's research is reviewed here because, in addition to his research being used to model harmonic constraints for melody generation (as part of the MOTIVE tool described in Chapter 7), the use of his *Pitch Space* model is also proposed as a method of defining a simple form of harmonic analysis (a necessary part of the formalisation process and development of the parser for the *Implication-Realisation Model*, described in Chapter 4).

2.5.1. Levitt's work on musical style

The constraint-based extensions to the parser (described in Chapter 7) are based on the work of David Levitt's (1984, 1985) hypothetical constraint language for musical dialects, and on an implementation of Levitt's proposals (Smith 1990). Many of the features of Levitt's constraint language have been simply described in the logic-based language Prolog (the language chosen for the implementation of the MOTIVE tool described in Chapter 7).

Levitt describes a constraint-based representation for musical style. In addition to describing constraint-based representations of many musical primitives and a number of music styles (such as "Bebop" and "ragtime"), Levitt uses his constraint-language to assist in the description of aspects of musical structure. In effect Levitt's work makes explicit many of the constraints necessary for creating artifacts for musical domains — metric and harmonic constraints are defined, in addition to melodic attributes such as contour and scalewise movements

(trajectories) towards key points in a piece.

Although, as stated by Levitt in his critical review, the level of sophistication of the representations and generations from his work is quite low, his system demonstrates how many elements of music can be expressed in constraints, and then used for the generation of plausible pieces.

Many of the constraints for metre and harmony that make up MOTIVE are based on those developed by Levitt. Perhaps the two most important contributions from Levitt's work to the design of MOTIVE are:

- **(general)** a view of the music composition process as being the definition of a set of constraints followed by the generation of pieces that satisfy the constraints, and
- **(specific)** a strictly hierarchical representation of metre, and the resulting power for both generation and measurement of the metric importance (metric strength) of note onset times.

2.5.2. Lerdahl's Pitch Spaces

Lerdahl (1988) suggests a model of Pitch Spaces that treats pitches, chords and regions in a single framework. His work is reviewed here since his pitch space framework is used as the basis for both a measurement of harmonic consonance and dissonance for our formalised version of Narmour's theory (see Chapter 4), and also for harmonic constraints in our tool for melody generation (Chapter 7). In this brief review, a summary is presented of Lerdahl's arguments for how his new form of tonal pitch space overcomes some problems in existing models of tonal pitch space, followed by a description of his pitch space framework and why it has potential for use in modelling harmonic concepts for melody analysis and generation.

Motivation for pitch space framework

Lerdahl's pitch space framework is not based on a symmetrical topological model (unlike those such as Longuet-Higgins (1962) and Shepard (1982)) — Lerdahl states that a weakness of topological models is too much symmetry (therefore misrepresenting the non-symmetrical aspects of the diatonic system), and claims his framework overcomes this. Lerdahl's pitch spaces are also able to model more than one level of pitch description in a single framework;

previous systems have modelled either *pitch classes* (e.g. Balzano 1982, and Shepard 1982) or tonal *regions* (for example the spaces of the eighteenth century and Weber 1824 / Schoenberg 1911/1978). One existing model that does model multiple levels of pitch description is that of Longuet-Higgins (1962/1987), but his system is applied to tonal regions (although derived from pitch classes) and cannot be used to describe pitch proximity. Work which has included descriptions of pitch classes, chord spaces and tonal regions was influential in the development of Lerdahl's pitch spaces (work cited by Lerdahl includes Krumhansl 1979 & 1983, Krumhansl, Bharucha & Kessler 1982, and Krumhansl & Kessler 1982). Lerdahl also states that since there is no a priori requirement that concepts of pitch, chord and key distance be modelled topologically, it is reasonable to suggest a non-topological framework which offers high explanatory power.

Description of Lerdahl's pitch space framework

Suggested by the ideas of chromatic, diatonic and triadic pitch "alphabets" overlearnt by listeners (Deutsch & Feroe 1984, Deutsch 1982b), Lerdahl's framework is a hierarchy of five spaces. The hierarchy is such that each level is made up of a subset of those pitch classes from the level immediately below. The five spaces are listed in Figure 2.3 below.

level of space	space name
a	octave space
b	open fifth space
c	triadic space
d	diatonic space
e	chromatic space

Figure 2.3: The 5 pitch spaces in Lerdahl's framework.

Three important points Lerdahl makes about the spaces are as follows:

- except for the chromatic space, the spaces describe the asymmetric patterns appropriate for diatonic music,
- the diatonic scale is ***directly*** represented in the framework (unlike in the symmetrical systems mentioned above), and

- the pitch space framework allows unified treatment of pitch class, chord and regional proximity.

Lerdahl uses Roman-numeral notation of chord / region, where the region numeral is in bold — for example, I/**I** is the pitch space for the tonic chord (C major) of the region C major. Figure 2.4 below shows both the pitch classes and alphabetic and numeric forms for the pitch space of I/I[5].

space	alphabetic pitch classes	numeric pitch classes
a octave space	C	0
b open fifth space	C G	0 7
c triadic space	C E G	0 4 7
d diatonic space	C D E F G B	0 2 4 5 7 9 b
e chromatic space	C D♭ D E♭ E F F# G A♭ B	0 1 2 3 4 5 6 7 8 9 a b
	embedding distance	0 4 3 4 2 3 4 1 4 3 4 3

Figure 2.4: Alphabetic and numeric pitch spaces for I/I[6].

The "embedding distance" in Figure 2.4 is a measure of how far from the octave space a given pitch class is for a given pitch space (this embedding shifts for a given chord and region). The shallower the embedding, the more important the pitch class harmonically for a given space. Another way of looking at a space is to count the number of times a pitch class is present (at all levels), the more times it is present the more important the pitch class.

Measuring the embedding distance is a vertical measure. Measurement horizontally Lerdahl explains as follows in terms of "skip" and "step":

> *"In traditional usage a step occurs between adjacent members of the chromatic or diatonic scales (a chromatic or a diatonic step), and an "arpeggiation" takes place*

---

[5] The use of pitch class 0 for C, 1 for C# or D♭ etc. is arbitrary, as is the choice of I/I. These choices have been made following Figures 4 and 5 of Lerdahl's (1988, p. 312).

[6] For notational convenience we have followed Lerdahl's convention of using "a" and "b" for the pitch classes of 10 and 11.

> *between adjacent members of a triad. It is more illuminating, however, to think of an arpeggiation as stepwise motion in triadic space. In similar fashion, one can speak of a step in open-fifth space or octave space. A leap of two octaves, on the other hand, is a skip in octave space. In sum, a step is adjacent motion along any level of the hierarchy, and a skip is nonadjacent motion — two or more steps — along any level."* (Lerdahl 1988, pp. 321-322)

Therefore, using Lerdahl's definition of step and skip, the proximity of two pitches in a given pitch space (e.g. I/II) can be measured as a "step distance" by the number of steps at a given level (either left or right) to get from one pitch to another (e.g. from p0 to p4 is one step in triadic space, two steps in diatonic space and four steps in chromatic space).

Chord proximity can be calculated using two factors: the diatonic circle of fifths and the number of common tones between the two chords. Lerdahl describes how each of these factors can be modelled via his pitch spaces. Lerdahl presents the chord-circle rule, defined as instruction to: *"move the pcs* [pitch classes] *at levels **a-c** four diatonic steps to the right or left (mod 12) on level **d**"* (p. 323). Thus there is no change to the diatonic or chromatic spaces when modelling the chord circle. The circle of fifths **[pc0 (I) - pc7 (V) - pc2 (ii) - pc9 (vi) - pc4 (iii) - pcb (vii) - pc5 (IV) - pc0 (I) and so on]** appears as a sequence of pitch spaces when the chord circle rule is successively applied, as illustrated for pc7 and pc2 in Figure 2.5 below.

space	V/I	ii/I
a octave space	7	2
b open fifth space	2 7	2 9
c triadic space	2 7 b	2 5 9
d diatonic space	0 2 4 5 7 9 b	0 2 4 5 7 9 b
e chromatic space	0 1 2 3 4 5 6 7 8 9 a b	0 1 2 3 4 5 6 7 8 9 a b

space	vi/I
a octave space	9
b open fifth space	9
c triadic space	0 4 9
d diatonic space	0 2 4 5 7 9 b
e chromatic space	0 1 2 3 4 5 6 7 8 9 a b

Figure 2.5: Construction of diagtonic circle of fifths with chord-circle rule[7].

The idea of steps and skips can be extended in terms of the number of applications of the chord-circle rule to move from one chord to another, a skip being when the rule needs to be applied more than once (so from "I" to "V" is a step, but "I" to "ii" is a skip, in the "I" region). Common tones between chords can be found (in levels *a* - *c*), for example the V/I space in Figure 2.5 above shares the pc7 with the space for I/I. However, the hierarchical position of common tones is also important (for example, "vi" shares more common tones with "I" than does "V", however it is the root of "V" that is common). Lerdahl defines the common tone distance between two chords as follows:

> *"... the common tone distance between two chords depends on the number of distinctive pcs between the two. In order to include the differences in weight of the root, fifth and third in the new chord, its distinctive pcs must be counted at all levels. As the underlined numbers in Figure 7* [our Figure 2.5] *indicate, V has four*

[7] This figure based on Figures 7a, 7b from Lerdahl (1988, p. 323).

distinctive pcs in relation to I [and] *ii has six distinctive pcs"*

(Lerdahl 1988, pp. 324)

The measure of distinctive pcs results in a value of 4 for both "vi" and "V", showing how such a calculation models the importance of the shared root of "I". The two measures (distance on circle of fifths and number of distinctive pitch classes) are useful when examining the relationship between chords, for example the distance from "I" to "V" would be (1, 4), and from "I" to "ii" (2, 6), from "I" to "vi" (3, 4).

Lerdahl suggests a simple overall measure of distance between chords, based on the simple summation of the two measures:

d	=	***j***	+	***k***
chord distance =		shortest no. of steps on circle of fifths	+	no. of distinctive pcs

Examples of chord distances include: d(V) = 1 + 4 = 5, d(ii) = 2 + 6 = 8, d(vi) = 3 + 4 = 7. This measure has transpositional invariance (so in region I for example, with I/I = (0 + 0) the distance from "I" to "vi" is 3 + 4, with vi/I = (0 + 0) the distance from "vi" to "IV" is also 3 + 4)[8].

A common tone circle can be formed by changing the numbers of steps in the chord-circle rule from 4 (a fifth) to 2 (a third). The circle would be: **[pc0 (I) - pc4 (iii) - pc7 (V) - pcb (vii) - pc2 (ii) - pc5 (IV) - pc9 (vi) - pc0 (I) and so on]**. Lerdahl creates a torus by the projection of the circle of diatonic fifths with this common tone circle, where stepwise motion takes place between adjacent chords. He points out: *"Because of* **d***'s transpositional invariance, these steps can be made equidistant from each other, disregarding the major or minor quality of any particular third motion. ... This equidistance is quite different from Longuet-Higgins' major-thirds vector"* (p. 325).

[8] The zeroing is important since it states from which position on the diatonic circle of fifths one begins counting, and from which pitch space distinctive pitch classes are counted.

Lerdahl points out that while the *d* formula is a simple addition of two very different things, it does lead to attractive theoretical results (he also goes on to show how it also correlates with experimental data).

The final level of description Lerdahl uses his pitch space framework for is to measure chord proximity across regions. Changes in the diatonic collection can be made with his *region-circle rule*: "*move the pcs at level* **d** *seven chromatic steps to the right of left (mod 12) on level* **e**". A step on this region circle causes a single pitch class to change. Repeated application of this rule gives a chromatic circle of fifths (the pitch classes are the roots of each diatonic space): **[pc0, pc7, pc2, pc9, pc4, pcb, pc6, pc1, pc8, pc3, pca, pc5, pc0 and so on]**. A new measure, *i*, can be added to the overall distance calculation for a chord, which is the number of steps on the region circle (which can be done simply by counting the differences in sharps and flats in the key signature). The calculation for *j* becomes an addition of the distance from the first chord to the new tonic, and then back from it. Lerdahl gives the following example: "*For example, for iv/vi, the value for j is not 2, as it would be directly from I/I (iv/vi = ii/I). Instead, j = 3 + 1 = 4; that is, three circle steps up to 'vi' and one back to 'iv'* " (p. 328).

Lerdahl goes on to present a measure of regional proximity, and also discusses how seventh chords and the minor modes can be treated in the pitch space framework.

Summary of review of Lerdahl's pitch spaces

The hierarchical and numerical nature of the pitch spaces, and the simplicity of the measurement of pitch, chord and region proximity suggest the use of this framework for computational modelling requiring a representation of the harmonic importance of given pitches (and their corresponding pitch classes). This pitch space formalism has very good explanatory power, and as Lerdahl goes on to discuss, appears to correlate to experimental results investigating pitch class stability (Krumhansl 1979, and Krumhansl & Sheperd 1979), multi-dimensional scaling of diatonic triads (Krumhansl, Bharucha and Kessler, 1982), and abstract region spaces (Krumhansl, and Kessler, 1982).

2.6. Conclusions from literature review

As discussed in the previous chapter, the research in this thesis is multi-disciplinary in nature. There are many books and publications on melody, and melody composition, however very few

of them consider melody analysis and the process of composition from the psychology of cognition perspective. Detailed explanations of the processes of composition (and why certain musical forms and pitch relationships are considered successful in melodies) are non-existent. Since no formal theories of composition are available, the research in this thesis has been led by the need to formalise the most promising analytical theory for melody — Narmour's *Implication-Realisation Model*. Although the theory is not without its critics, (see the detailed discussion of published reviews in the next chapter) it has a number of strong attributes in its favour: it is grounded in psychological theories of cognition and perception; it is extensively described in Narmour's publications; it is a model of intrinsically melodic theory rather than harmonic; it is expressed in terms of declarative sub-theories of how listeners process the different attributes of melodic events; and finally, it is hierarchical and can be recursively applied, providing a strong basis for the use of the theory in a constraint-based intelligent learning environment for melody *composition*.

Chapter 3:
Narmour's theory for the analysis of melodies

3.1. Chapter outline

This chapter presents a description of Eugene Narmour's *Implication-Realisation Model*, taken from Narmour's publications about the theory (Narmour 1989, 1990, 1992). The chapter breaks down into three main stages:

- a presentation of Narmour's theory (§3.2 to §3.9),
- a review of published critiques of the *Implication-Realisation Model* (§3.10),
- a discussion of the psychological validity and strengths of the theory (§3.11).

The first stage is a presentation of Narmour's theory without criticisms. Chapter 4 critically discusses inconsistencies and limitations of the theory, and presents a set of simplifications, extensions and formalisations. It is the formalised version of the theory described in the next chapter that has been used as the basis for the computational parser described in Chapter 5.

3.2. Implications rather than expectations

The *Implication-Realisation Model* gets its name from the way low level perceptual processes generate *implications* generated note-by-note as a listener hears a melody. Narmour proposes two low level rules of inference describing what implications are generated at a given point, from the melodic parameters of a sequence of notes. The way groups of heard notes are classified (and what form the boundaries of note groupings) is determined by the extent to which the generated implications are *realised* by following notes (i.e. which implications for each melodic parameter are

met or denied by the subsequent notes).

Narmour makes it very clear that the universal primitives he proposes deal with the *implications* generated note-by-note, these low level predictions are to be distinguished from *expectations*, since the rules of inference in his theory model the lowest cognitive level which does not have access to the stylistic knowledge at the higher levels about which the listener can introspect[1]. Narmour's theory has two elements, those modelling the *bottom-up* processing of notes in terms of implications and realisations/denials, and the second element being how a listener's learned knowledge of style influences the cognitive processing of heard musical events from the *top-down*. It is the bottom-up aspects of Narmour's theory that are the basis for the work in this thesis, although some of the issues involved in the formalisation of the top-down stylistic processes are addressed in the further work section of Chapter 8. In additional in the previous chapter some parallels have been drawn between Narmour's discussion on style and various constraint-based models of music perception and processing.

3.2.1. Narmour's two rules of inference

The *Implication-Realisation Model* has two rules of inference (which Narmour calls *hypotheses*), that state generally what implications will result from a sequence of two notes (two notes are needed because the rules of inference are expressed in terms of the size and contour of an *interval* between two notes).

Narmour's theory claims that low level perceptual structures of melody rest on the realisation or denial of the following two inference rules, and that such structures exhibit either closure or nonclosure (a later section in this chapter discusses precisely what Narmour means by closure in terms of the *Implication-Realisation Model*).

The implications are in terms of the melodic parameters of *interval* and *contour*. The size of the interval generating the implications is the factor determining which rule of infer ence is in effect. Whether an interval is large or small is measured on a "parametric scale". The parametric scales for

---

[1] An important feature of the bottom-up aspects of the *Implication-Realisation Model* is that it is modelling automatic, low level perceptual processes, which are not influenced by higher cognitive levels (although the final analysis of the melody is indeed influenced by top-down stylistic knowedge).

the different parameters of melody modelled by the *Implication-Realisation Model* are described later this chapter.

3.2.2. The rule of continuation (for small intervals)

> *"When form, intervallic patterns, or pitch elements of a given melody are similar, the listener subconsciously or consciously infers some kind of repetition of pattern, element, or form."* (Narmour 1990, p. 3)

The rule of continuation is in triggered when the interval generating the implication is small. It could be restated as follows: if by some measure of similarity, two contiguous melodic features are similar (for a given parameter), then there is a perceptual implication that the value of the same parameter for the next melodic event will also be similar. Narmour notates this rule of inference for form, intervallic patterns and pitch elements as follows ("+" notates two contiguous events in chronological sequence, "→" notates perceptual implication):

- form (italic capitals), $A + A \rightarrow A$,
- intervallic elements (capitals), $\mathrm{A} + \mathrm{A} \rightarrow \mathrm{A}$,
- pitch elements (lower case), $\mathrm{a} + \mathrm{a} \rightarrow \mathrm{a}$.

The rule of continuation is based on the Gestalt psychological laws of common fate and proximity (Koffka 1922 and 1935, and Katz 1951).

The two parameters on which the *Implication-Realisation Model* focuses are interval size (between two pitches), and melodic contour[2] — so, for example, an ascending, scalewise sequence such as that in Figure 3.1[3] would result (after the first two notes) in implications of continued small intervals and ascending contour, these implications (for both parameters) are realised for all the notes in the figure.

---

[2] Narmour uses the term "registral direction" when referring to melodic contour — in both cases the relative direction (ascent, descent or lateral) between two notes is being referred to.

[3] In Figure 3.1 we have used a diatonic scale for our example, which has a mixture of interals of one or two intervals (both of which are always considered small by the *Implication-Realisation Model*). A chromatic sequence could just as easily have been used.

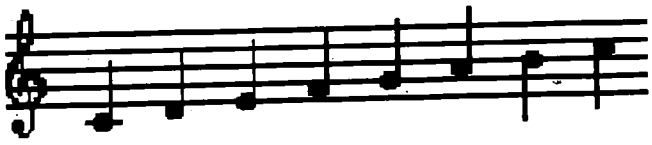

Figure 3.1: An example of realised implications of continuation.

3.3.2. *The rule of reversal (for large intervals)*

> *"When form, intervallic patterns, or pitch elements of a given melody are different, the listener subconsciously or consciously infers some implied change in form, pattern, or element."* (Narmour 1990, p. 3)

The rule of reversal is in triggered when the interval generating the implication is large. This could be restated as follows: by some measure of similarity, two contiguous melodic features are different (for a given parameter), then there is a perceptual implication that the value of the same parameter for the next melodic event will also be different:

- form (italic capitals), $A + B \rightarrow C$,
- intervallic elements (capitals), $\mathrm{A} + \mathrm{B} \rightarrow \mathrm{C}$,
- pitch elements (lower case), $\mathrm{a} + \mathrm{b} \rightarrow \mathrm{c}$.

The rule of reversal is based upon Narmour's own musical intuitions, and similar ideas can be found in theories of others, for example Meyer (1956); however, Narmour does admit that there is not (as yet) psychological justification for the rule of reversal, and proposes it as a "symmetrical" construct to the (psychologically supported) rule of continuation.

Figure 3.2 illustrates a melody fragment where the rule of reversal results in (realised) implications of change, for both the parameters of interval and of contour — the large interval from the C to A is followed by the small interval from A to G, the ascending first interval is followed by a descending interval.

Figure 3.2: An example of realised implications of reversal.

3.3. The grouping of notes (introduction to structures)

Contiguous notes are grouped when they all realise or deny the same implications. Implications for a sequence of contiguous notes are initiated by the magnitude of the first interval in the sequence — Narmour states that if this first (*generating*) interval is small, then the inference rule of continuation applies, and the implication for succeeding intervals will be for similar small intervals, and a similar contour; if, on the other hand, the generating interval of a sequence of notes is large, then the inference rule of reversal applies, and the implications will be for the next interval to be small, and its contour to differ from that of the first interval. Whether the generating interval is considered large or small is determined by Narmour's proposed "parametric scales", explained later this chapter.

Note that groups of notes which have implications generated by the inference rule of continuation have no obvious end to the sequence, because each new note (as well as realising the implications from the preceding interval) generates new implications of the same form (i.e. future notes that realise such implications could all be considered part of the same group — hence the sequence in Figure 3.1 of eight notes, which could continue). However, groups of notes which have implications generated by the inference rule of reversal can only contain three notes, since once a (large) generating interval is followed by a different, small interval, the small interval will generate different implications to that of its predecessor — thus there are no additional notes that could be added after the G in Figure 3.2 that would also be included in the same reversal group[4].

Since the *Implication-Realisation Model* states that implications are formed for the two melodic parameters of interval size and contour, and that an implication can be either realised or denied, eight combinations of implication and realisation can occur — these are summarised in Figure 3.3 below. The two columns "int" and "cont" refer to the implications for the parameters of interval and contour respectively.

[4] Of course, according to the hypothesis of continuation, the second, small, interval in a reversal structure will generate its own implications of contiuation. Under certain conditions structures can merge together, and in some cases the second interval of a reversal structure can form the generating interval for a continuation structure. This is discussed in detail later this chapter.

small generating interval (rule of continuation)

int:	realised	cont:	realised
int:	realised	cont:	denied
int:	denied	cont:	realised
int:	denied	cont:	denied

large generating interval (rule of reversal)

int:	realised	cont:	realised
int:	realised	cont:	denied
int:	denied	cont:	realised
int:	denied	cont:	denied

Figure 3.3: Possible combinations of implication and realisation.

Narmour calls a sequence of contiguous notes that realise and deny the same implications a *structure*. Later in this chapter (section 3.6) the different kinds of structures are described in detail. Each of the above eight possible combinations of large or small interval, and realised or denied parameter of interval and contour forms part of the definition of one or more classes of note grouping.

3.4. Closure

A key concept of the Implication-Realisation Model is that of ***closure***. It is closure that causes a structure to terminate, determines how a terminated structure relates to the next structure for a level, and whether the initial and terminal notes of a structure are promoted in the analysis hierarchy.

Narmour uses the term ***closure*** in a general sense, where closure varies in degree of strength, occurring throughout a melody, as well as at points when a number of melodic parameters combine in a manner indicating a summation. Narmour defines his use of the term as follows:

> Closure refers *"... to the various ways musical parameters interact to create melodic 'chunks', perceptual groupings whose beginning and ending notes exhibit varying degrees of stability..."* (Narmour 1990, p. 45)

> *"By closure I refer to syntactic events whereby the terminating, blunting, inhibiting, or weakening of melodic implication occurs."* (Narmour 1990, p. 102)

Strong closure causes structures to terminate, and if closure on a note is strong enough the note is promoted to the next hierarchical level of the analysis. Closure is caused by melodic parameters (such as short to long durations of notes), and also from other musical parameters, such as metre (weak beat to strong beat) and harmony (dissonance to consonance). It is a combination of the two rules of inference and closure that determine how notes are grouped (i.e. which structure they belong to) and where the end of each structure occurs. Closure is the way the theory models the low level influence of melodic parameters in addition to interval and contour (i.e. duration). Closure can also be stated in terms of the top-down influence of style, this is the case when closure is attributed to metre or harmony.

Perhaps the most obvious form of closure (which Narmour calls "stopping"), and which always terminates a structure, is when a new note is heard which implies and realises melodic parameters in a different way to the ongoing structure (for example a large interval during a structure where the implications are for small intervals). If closure is weak, structures may merge in a number of ways, such as sharing a note (the last note of one structure being the first of another), or the sharing of two notes (for example the last two notes of a reversal becoming the first two notes of a processive structure). The merging of structures is described later in this chapter.

Narmour states that closure occurs in varying degrees — he draws a parallel between his notions of closure and those of Tenny and Polansky (1980). Narmour's theory categorises closure in the following three ways:

articulation weak closure,

formation closure that is almost strong enough to generate a new hierarchical level, but does not actually create one,

transformation (structural transformation) closure strong enough to cause the note on which it occurs to be promoted to the next hierarchical level.

Narmour lists six cases where closure occurs (Narmour 1990, p. 102):

(1) when simple stopping takes place,

(2) when metric emphasis is strong,

(3) when consonance resolves dissonance,

(4) when duration moves cumulatively (short note to long note),

(5) when intervallic motion moves from large interval to small interval, and

(6) when registral direction changes.

3.4.1. Closure due to stopping

Stopping is a form of retrospective closure, i.e. it is caused by a note that is a candidate to join a structure, but causes the structure to be terminated. The three causes of stopping closure are:

- end of the melody (in which case the last note of the melody may be promoted, according to other closure upon it, and any interrupted style implications),
- a significant rest,
- a note which denies the implications of a continuation structure.

In all cases the current structure is terminated, but whether notes are promoted, or any merging occurs between structures is dependent on other forms of closure also happening on the terminal note of the structure.

Durational (prospective)

Durational closure occurs when the most recent note to be added to a structure has a duration greater than the preceding note. This situation of a sequence of two notes of increasing duration Narmour calls "cumulative" duration. According to the *Implication-Realisation model* significant cumulative durational closure is always transformational (i.e. leads to the promotion of the terminal note of the structure having the long duration). Narmour refers to the closure due to duration as "melodic suppression", the suppression is due to the weakening of implications over time, so when the implied note does not begin the same duration after the previous note, implication begins to fade. The longer the delay after the implied onset, the stronger the closure due to duration. Narmour cites a number of pieces of psychological research to support the plausibility that such "updating" of implications can occur (Narmour, 1990, p. 109, citing Dowling and Harwood 1986, and Massaro 1972)[5].

[5] Narmour (1990, p. 107) also cites Fraisse (1982), Mursell (1937) and Woodrow (1911), stating that in many cases durational cumulation and metric emphasis often occur at on the same notes. He goes on to say that in many of his analyses he only notates the closure due to duration.

Narmour describes two cases for the amount of closure due to cumulative duration[6]:

releasable suppression — a temporary release in implication due to the cumulative duration, but not sufficient closure to cause the structure to be terminated,

definition: $D2 < (D1 * 1.5)$

nonreleasable suppression — cumulative duration leading to transformational closure,

definition: $D2 \geq (D1 * 1.5)$

Closure in reversals - R

Narmour presents a number of rules describing how to measure closure of reversal structures (p. 158, Narmour 1990):

(1) Formal rule:

a. the greater the intervallic differentiation, the greater the closure.

(2) Function rules:

a. initial interval: the stronger the implication (i.e. the larger the interval), the stronger the closure;

b. terminal interval: the weaker the implication (i.e. the smaller the interval), the stronger the closure[7].

(3) Serial-position rule:

a. the weaker the implication of the terminal interval (the smaller the interval), the stronger the closure.

3.4.2. Metric closure

Metric closure is prospective (i.e. occurs on a note, rather than due to a following note or rest), and occurs when a note has an onset time that is metrically more important than that of the preceding note.

---

6 The symbols "D1" and "D2" refer to the length of duration of the two notes under inspection, "D2" being the most recent note to be added to the structure experiencing duational closure. The threshold of 1.5 times duration is defined by Narmour (1990, p. 108).

7 This follows from the fact that the small terminal interval of a reversal generates implications of similarity for more small intervals (following the application of the hypothesis of continuation).

3.4.3. Harmonic closure

Harmonic closure is prospective (i.e. occurs on a note, rather than due to a following note or rest), and occurs when a note is more dissonant that the preceding note.

3.4.4. Stylistic closure

Closure due to the top-down influence of style is discussed generally by Narmour (1990, 1992). However, we have not attempted to model stylistic closure since the amount of research required to investigate that aspect of Narmour's theory is beyond the bounds of the research described in this thesis. It is useful to very briefly describe the two classes of stylistic influence defined by Narmour, since they form the basis for some of the work described in the further research section in the final chapter of this thesis:

intra-opus this is style that is local to a single piece; i.e. conventions adopoted earlier in a piece will influence top-down expectations when similar events occur later in a piece, and

extra-opus other style, that has been learnt from previously heard pieces of music (and to which, presumably, intra-opus style contributes once a piece has finished sounding).

3.5. Narmour's parametric scales for melody

As we have already noted, the determination of whether a generating interval is large or small, and of whether notes succeeding the generating interval exhibit similarity of interval and contour, is through the use of what Narmour calls "parametric scales". Narmour uses parametric scales to model the cognition of the melodic parameters of interval and registral direction[8]. These three scales are described below.

3.5.1. Interval

Figure 3.4 shows Narmour's parametric scale for interval (the top row of numbers 0..14 are the interval sizes expressed in semitones, the second row of letter-number pairs (e.g. "m2") are

---

[8] Narmour also proposes the use of such scales for other parameters, such as duration, dynamics and tempo (1990, p. 287), and harmony (p. 288) — these proposals are not yet part of Narmour's *Implication-Realisation Model*, and are not considered in this thesis.

abbreviated names for intervals). As can be seen, the scale is divided into three regions: small, threshold, and large intervals.

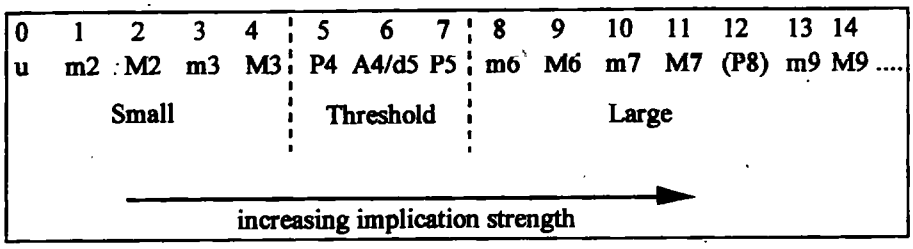

Figure 3.4: Narmour's intervallic parametric scale.

As can be seen in the figure, intervals of a major third and less are considered small, those from a perfect fourth to a perfect fifth are threshold values[9] (i.e. in certain circumstances could be considered small or large[10]), and intervals a minor sixth and above are considered large. Note that the octave (perfect eighth — "(P8)") is a special case as described below.

The octave — a special interval

The octave is a special interval, since two notes with one or more octaves between them are often considered the same — in Schenkerian analysis (Schenker, 1979) the invocation of "octave transfer" is often used to reduce intervals (for example a ninth to a second)[11]. In the *Implication-Realisation Model* "octave transfer" is only used with reference to intervals of a whole number of octaves.

The octave is especially interesting in the *Implication-Realisation Model*, because if treated as a large interval (because it is larger than a perfect fifth), it would trigger the inference rule of reversal; however, if octave transfer is considered to have occurred in the listener's reading of the interval, then the interval is unison, and so the inference rule of continuation is applied. Although Narmour does provide some description of the special cases for the octave, in our work on the formalisation of his

---

9 Narmour appears to have extended the size of the threshold, since in his article on the "Genetic Code of Melody" (Narmour 1989, p. 51) only the tritone (A4/d5) was considered the threshold. The scale above is from Narmour's book (1990), which is considered the definitive version of his theory for the research described in this thesis.

10 Narmour cites Balzano and Liesch (1982) for a discussion of the ambiguity of the tritone. Since Narmour also mentions that perfect fourths are rarely confused with perfect fifths, it is perhaps odd that the threshold has been extended to include them (see previous footnote).

11 The reference to octave transfer in Schenkerian analysis is from Narmour (p. 233, 1990).

theory we have made the simplifying assumption that an octave always counts as unison, therefore we do not present Narmour's special treatment of the octave in this chapter.

3.5.2. Registral direction (contour)

Figure 3.5 shows Narmour's scale for registral direction. As can be seen, only lateral-to-lateral contour is considered "sameness", however any two intervals with the same contour (lateral, ascent, or descent) are defined as being similar by the parametric scale. When there is differentiation of contour closure occurs to some degree.

(sameness)	*(similarity)*	*(differentiation)*
lateral to lateral	ascent to ascent	ascent to descent
	descend to descent	descent to ascent
		ascent to lateral
		descent to lateral
		lateral to ascent
		lateral to descent

Figure 3.5: Narmour's parametric scale for registral direction.

3.6. Structures

Narmour describes a set of "structures" (resulting from the rules of continuation and reversal), these structures group together contiguous notes in a melody which realise and deny the same expectations. Note that the structure list is exhaustive, i.e. every note in a melody will be a member of one of the given structures.

Narmour's descriptions of the definitions of some of these structures is somewhat obscure (as discussed in section 4.1 in the next chapter), therefore in this section for clarity of exposition we have used the results of our clarification to some extent to present the definitions in a straightforward way.

Figure 3.3 has been repeated below as Figure 3.6, but with the addition of the names of the most common structures. The structures are named with the a combination of the letters [**D, P, I, V, R**], the meaning of which is described shortly.

small generating interval (rule of continuation)

int:	realised	cont:	realised	[P] [D]
int:	realised	cont:	denied	[IP] [ID]
int:	denied	cont:	realised	[VP]
int:	denied	cont:	denied	

large generating interval (rule of reversal)

int:	realised	cont:	realised	[R]
int:	realised	cont:	denied	[IR]
int:	denied	cont:	realised	[VR]
int:	denied	cont:	denied	

Figure 3.6: Possible combinations of implication and realisation[12].

Figure 3.7 below illustrates a number of these structures using three note, artificial melodies.

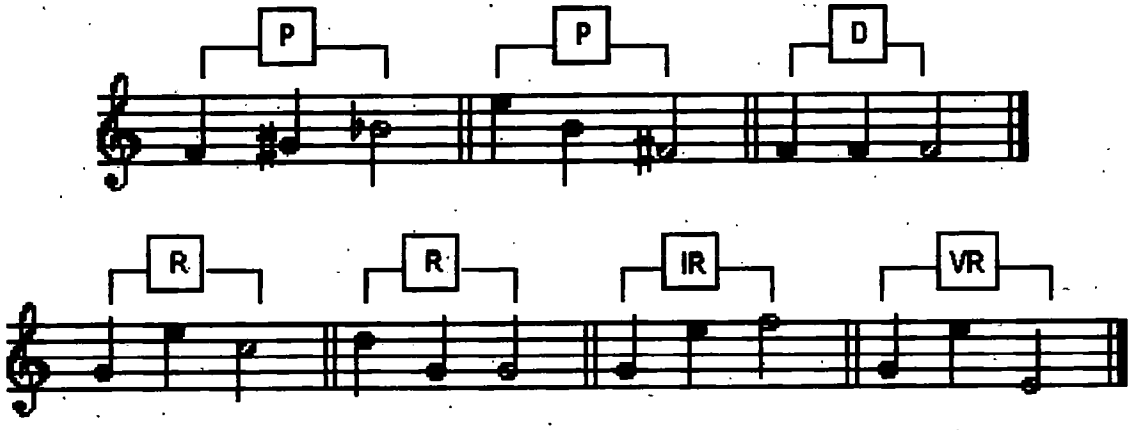

Figure 3.7: Examples of melodies resulting in different structures.

The two rules of inference lead to two classes of structures, continuation structures are either "Iterative" (where pitches or intervals are repeated) or "Processive" (where initial contour and interval size are continued). There is only one category of reversal structures. Since one needs two notes to create a "generating interval", and a third note to determine realisation or denial of implications, the smallest category for most structures is three notes. In cases where closure is very

[12] The table in Figure 3.6, and the related decision tree of Figure 3.8, have been developed as part of our clarification process, but are presented in this chapter to aid the exposition of Narmour's theory. These formalisms have been inferred from implicit states made by Narmour.

strong, implication may be stopped before three notes (of a new structure) have been heard, in which case one of two special structures occurs: *dyads* (two notes), and *monads* (a single note).

Thus the four classes of note groupings to be considered are:

- continuation,
- reversal,
- dyads (two note groups),
- monads (single note groups).

Narmour also considers a fifth type of note grouping, ***'registral return'***. Narmour distinguishes between *exact* and *near* registral return. Exact registral return is found in intervallic duplication **[ID]** structures (described below), describing the case when a pitch is followed by a different pitch, then the first pitch is repeated (i.e. the melody has *returned* to the original pitch). Near registral return is found in intervallic process structures **[IP]** (see below), in these structures the third note is very close to the first, but not exactly the same pitch. Both cases are realisations of the Gestalt law of proximity, in this case in the parameter of pitch. Narmour describes registral return as being part of a parallel cognitive process to the perception of the four types above, they appear to be used to illustrate relationships not represented by the main part of the *Implication-Realisation Model*. Registral return and near registral return are have not been considered necessary for the formalisation and modelling of Narmour's theory in this thesis.

Each structure has a *processive* and a *retrospective* form. The retrospective form describes cases when the initial implications of a structure are unexpectedly denied, and *retrospectively* a different structure than expected can be seen to have occurred. These retrospective structures are defined after the prospective forms below.

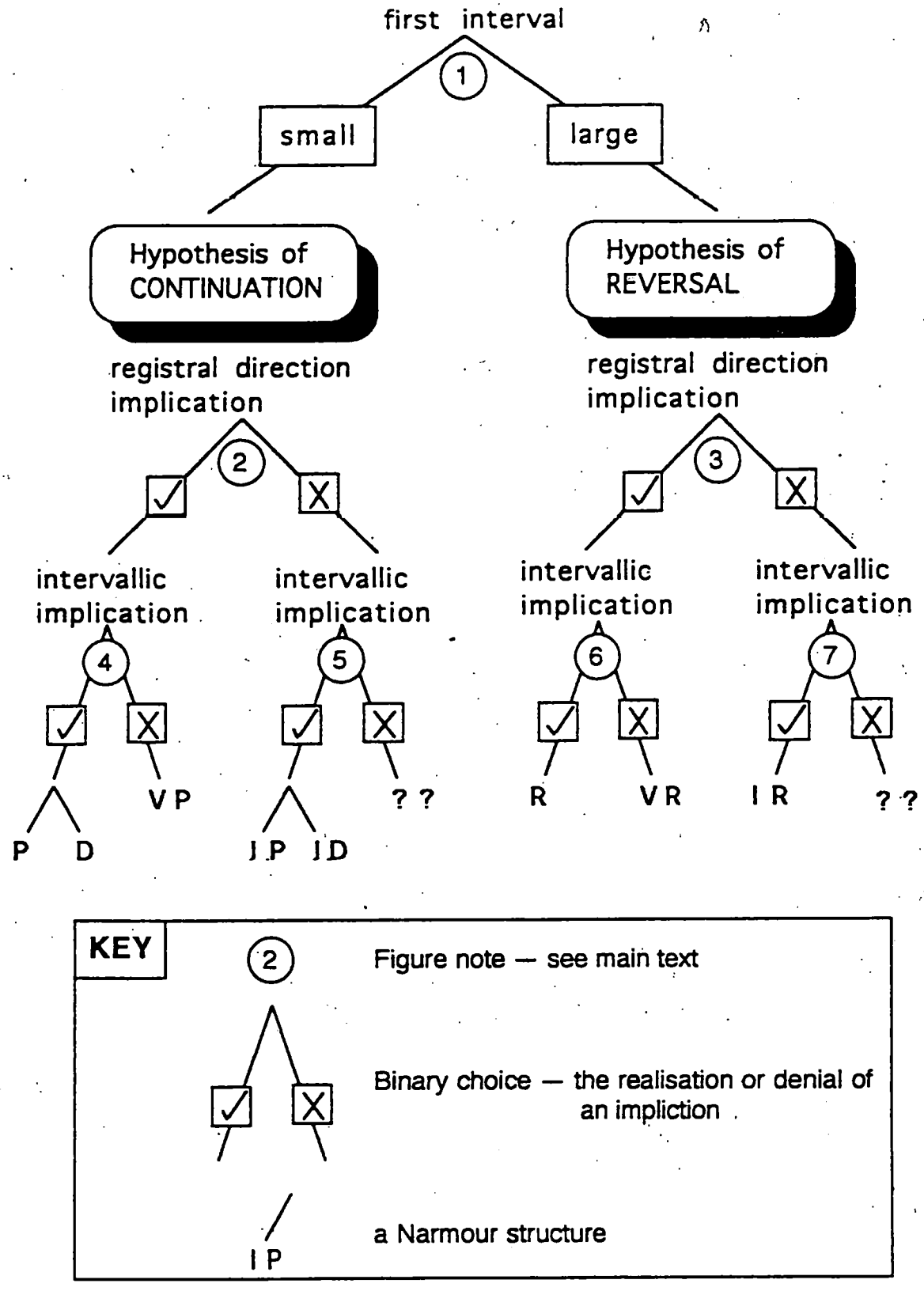

Figure 3.8: Graphical taxonomy of Prospective structures.

Figure 3.8 presents Narmour's taxonomy of structures in a graphical tree notation, this is essentially a graphical form of the table presented in figure 3.6. The figure should be read from the top,

downwards. This figure is essentially a decision tree for the determination of a structure given the interval magnitudes and contours for a sequence of notes. The figure has been derived from implicit statements of Narmour's, although he does not present such a decision tree himself.

Notes for Figure 3.8

(1) The definition of whether an interval is large or small is determined by Narmour's parametric scale for interval (as discussed earlier). Small intervals trigger the hypothesis (inference rule) of continuation, large ones trigger the hypothesis of reversal.

(2) If the hypothesis of continuation is active, the implication for the contour (registral direction) is for the second interval to have the similar contour (according to the parametric scale for contour) as the first. If this is the case the left branch is taken, otherwise the right branch is taken.

(3) If the reversal hypothesis is active, a different contour (registral differentiation, as defined by the parametric scale for contour) is implied for that of the second interval, if this is the case, the left branch is taken. If not the right branch is taken.

(4) If the implication for another small interval is realised, the left branch is taken and the intervals are classified as part of either a process structure, or a duplication structure (being a special case of process, where the interval is zero, and contour is lateral).

(5) If the implication for another small interval is realised, the left branch is taken and the notes are classified as part of either an intervallic process structure, or an intervallic duplication structure (being a special case of intervallic process, where the intervals is identical). The question marks indicate that if the initial interval is small, and neither of the implications of similarity are realised, the result is not a processive structure — later this section we discuss the retrospective structures, some of which fill this gap (i.e. retrospectively the initial implications are revised).

(6) Since to get to this choice point the hypothesis of reversal has been triggered, the implications are for differentiation in each parameter. If the next interval is small (i.e. differentiated from the large first interval), the structure is a reversal one; if the next is large, the structure is a registral reversal.

(7) Since to get to this choice point the hypothesis of reversal has been triggered, the

implications are for differentiation in each parameter. If the next interval is small (i.e. differentiated from the large first interval), the structure is an intervallic reversal one; if the next is large, then neither of the implications of differentiation have been realised, and no prospective structure occurs — again the definition of retrospective structures later in this chapter explain what fills this gap.

3.6.1. Continuation structures (Duplication and Process)

When the first interval of a structure is small the hypothesis of continuation is initiated, thus all prospective continuation structures begin with a small interval.

Process - P, IP, VP

All three process structures **[P]**, **[IP]** and **[VP]** begin with a small interval (thus the hypothesis of continuation is in force). Small intervals (for prospective structures) are initiated with intervals of a perfect forth or less.

A *process* structure **[P]** is a sequence of notes where all the intervals are small, and the contour does not change (e.g. a diatonic scalewise sequence). All intervals must be a perfect fourth or less, and the difference between any two contiguous intervals in the process must be no larger than a minor third (this is Narmour's definition of similarity of intervals when contour does not change.).

When the contour changes, but all intervals in the note sequence are small (a perfect forth or less), the structure is an *intervallic process* **[IP]**. However, the difference between any two contiguous intervals is reduced to a major second and below (since the theory states that the definition of similar intervals is narrower when contour changes). Note that intervallic process describes cases of near registral return, which Narmour symbolises as **[a b a']**.

When all intervals are small (a perfect fourth or less) and contour does not change, but there occurs an interval that differs by more than a major third from its, the structure is an *registral process* **[VP]**[13].

---

13 The "V" being a mnemonic for "Vector" (i.e. direction).

Duplication (iteration) - D, ID

A *duplication* structure **[D]** is when a pitch is repeated, thus both the intervallic and contour parameters are the same for all notes in the structure. A duplication structure must contain at least three notes (three notes are required to form the first two intervals of a duplication structure; "duplication" is the set of all intervals for a single duplication structure). Since all intervals are small (a perfect forth or less), and the contour does not change, duplication structures are a sub-set of process structures.

When the first interval of a structure is small (a perfect forth or less) and repeated, but the contour reversed, the structure is called *intervallic duplication* **[ID]** and thus leads to *registral return*, (when one pitch occurs, changes to a second, and then the first pitch is repeated). Note that intervallic duplications structures describe instances of exact registral return, which Narmour symbolises as **[a b a]**.

3.6.2. Reversal structures - R, IR, VR

All three prospective reversal structures are based on the hypothesis of reversal, which is initiated by the first interval of a structure being large.

A *reversal* structure **[R]** is a sequence of three notes, where the interval between the first two is large (a perfect fifth or more), and between the second two notes is smaller by at least a minor third, and contour changes.

When the intervals for a three note sequence are large then smaller, but contour does not change, an *intervallic reversal* **[IR]** has occurred. The first interval must be a perfect fifth or larger, and the second interval must be smaller, and differentiated from the first by a major third or more.

If the intervals are both large, but registral direction changes, then *registral reversal* **[VR]** - has occurred. For this structure the first interval must be a minor sixth[14] or more, and the second interval must be larger than the first.

---

[14] Also a diminished fifth can give rise to a registral reversal, but if an augmented fourth then *retrospective* registral reversal occurs **[(VR)]** (Narmour 1990, p.336).

3.6.3. Retrospective structures - (P), (IP), (VP), (R), (IR), (VR), (D), (ID)

Retrospective structures occur when the initial implication (of continuation or reversal) resulting from a small or large generation interval is not realised - for example an interval that ordinarily implies continuation retrospectively realises a reversal pattern.

Retrospective duplication - (D)

This structure seems intuitively impossible, since if a note is repeated three or more times then it is a (prospective) example of a duplication structure, and if a note is not duplicated three times, then no duplication has occurred. However, Narmour suggests (Narmour 1990, p. 277) a case where a particular type of melodic motive appears a number of times during a melody (thus creating intra-opus style expectations). If such a motive were a note repeated twice, and then the implied duplication broken in some way, when a similar melodic pattern is met later in the melody, the implication would be that after two notes the pattern's implications are broken. If the third note were to occur, it would be a surprise, and in such a situation, retrospective duplication would be said to have occurred.

Narmour also suggests that when, after hierarchical promotion, a duplication occurs, that the listener was not led to expect at the previous hierarchical level, then retrospective duplication would have occurred. Narmour goes on to state than consideration of how form affects high level implication is delayed until his (forthcoming) third volume on the Implication-Realisation Model.

From the point of view of definition, the definition for duplication above also defines the conditions for retrospective duplication.

Retrospective intervallic duplication - (ID)

When an initial large interval is followed by a sequence of intervals of different contour, and the second contour is different from the first, and when all intervals in the sequence are the same, a retrospective intervallic duplication **[(ID)]** structure has occurred. All intervals in such a structure need to be a perfect fifth or larger.

Retrospective process - (P)

When an initial large interval is followed by a sequence of intervals in the same contour, and when all contiguous intervals in the sequence are similar (i.e. differ by a minor third or less), a retrospective

process [(P)] structure has occurred. All intervals in such a process need to be a perfect fifth or larger.

Retrospective intervallic process - (IP)

When an initial large interval is followed by a sequence of intervals of with at least one with different contour, and the second contour is different from the first, and when all intervals in the sequence are the similar (defined as less than a major second in this case), a retrospective intervallic process [(IP)] structure has occurred. All intervals in such an intervallic process need to be a perfect fifth or larger.

Retrospective registral process - (VP)

When an initial large interval (or an initial interval of a diminished fifth) is followed by a sequence of intervals of the same contour, and when the intervals are in a sequence of increasing magnitude, with less than a major third between any two contiguous intervals, a retrospective registral process [(VP)] structure has occurred. All intervals in such a process need to be a perfect fifth or larger.

Retrospective reversal - (R)

In the case of retrospective reversal only an initial interval of a minor third, major third or perfect fourth can initiate the structure. Contour changes, and the second interval is smaller than the first by at least a minor third.

Retrospective intervallic reversal - (IR)

Only a perfect fourth can initiate a retrospective intervallic reversal. Intervallic differentiation occurs, with the second (smaller) interval being at a major third or more smaller than the first interval. Contour does not change.

Retrospective registral reversal - (VR)

When a structure begins with an interval of a perfect fourth or less[15], followed by a larger interval and a change of contour, a retrospective registral reversal has occurred.

---

15 An augmented fourth can also give rise to a retrospective reversal structure, but in the case of a dimished fifth, then the structure would be *prospective* registral reversal. This is also a case where the octave (P8) can play the role of a small interval, and so initiate a retrospective reversal structure.

3.6.4. Dyads - [interval size]

A dyad is a group of two notes where closure has occurred on the second note — thus the implications from the generating interval have no opportunity to be realised or denied. Dyads are notated by the number of semitones of the interval[16].

3.6.5. Monads - M

Monads are single notes that are non-implicative (for example due to being followed by a rest, thus there being no following note to form implications). Monad structures are notated with the symbol [M].

3.6.6. Examples of structures

Figure 3.9 illustrates the following structures:

- **P - Process**, small intervals, all contours the same,
- **IP - Intervallic Process**, small intervals, but contour changes,
- **VP - Registral Process**, not all intervals small (but first is), contours the same,
- **D - Duplication**, all notes the same,
- **R - Reversal**, initial interval large, followed by small interval and change of contour.

Note the graphical notation for indicating the initial and terminal notes of a structure by vertical lines, with the name of the structure (or structures) in the middle of the horizontal lines pointing inward for the terminal notes.

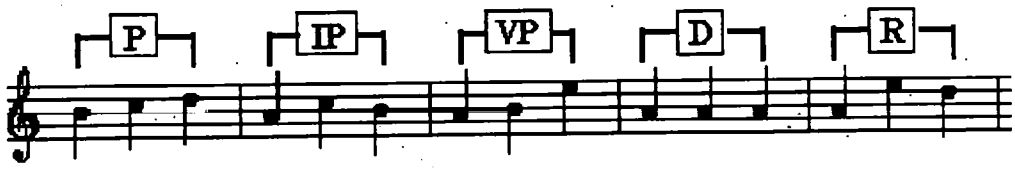

Figure 3.9: Some examples of melodic structures.

16. Narmour also discusses the use of numbers to indicate the size of the interval has a tradition in music notation - for example in figured bass.

3.7. Combination of structures

When (due to weak closure) two or more structures share intervals, a contiguous sequence of structures occurs, for which only the first note of the first structure, and the last note of the last structure in the sequence may be hierarchically promoted (hierarchy is discussed later in this chapter) — for an example of such a sequence see Figure 3.12. Although this is a single concept, Narmour distinguishes between cases where only two structures share an interval (Figure 3.12), which he calls combining, and cases where three or more contiguous structures share intervals (Figure 3.13), which he calls *chaining*.

There are five conditions that cause combining or chaining (Narmour 1990, p. 10):

(1) the occurrence of dissonance on a metric accent,

(2) the presence of ongoing metre,

(3) the envelopment of metric accent by process - **P** - in a context of *additive* (isochronous) or *countercumlative*[17] (long to short) durations,

(4) the envelopment of metric accent by duplication - **D** - in an additive context, and

(5) the envelopment of metric accent by a harmonic process in an additive context.

3.8.1. No notes shared

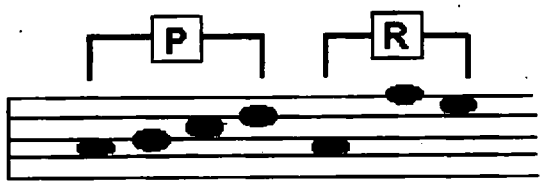

Figure 3.10 : No combination of structures.

When two adjacent structures share no notes, there is the potential for the terminal note of the first structure and the initial note of the second structure to both be hierarchically promoted. To indicate this situation of no note sharing, the "square brackets" of the two structures are notated so as not to touch. This is illustrated in Figure 3.10 above.

[17] The terms *additive* and *countercumulative* are Narmour's (1990, 1992).

3.8.2. *Shared note (joined)*

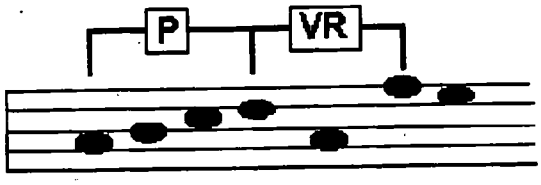

Figure 3.11 : Note shared between structures.

As can be seen in Figure 3.11 above, when two adjacent structures share a note (the terminal of one structure also being the initial note of the second), the two structures are "joined". This joining is indicated by the square brackets touching over the shared note. The consequence of joined structures is that one less note may be promoted than in the case of no shared notes (since the terminal note of the first structure is also the initial note of the second structure). In Figure 3.11 above, the fourth note is shared between the process [P] structure and following registral reversal [VR] structure.

3.8.3. *Combination (merging)*

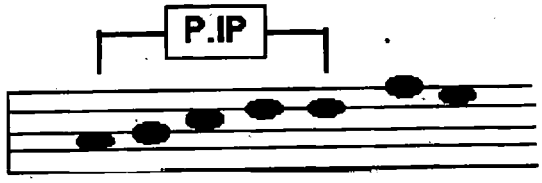

Figure 3.12 : Combination of two structures.

When closure is very weak on the terminal note of a structure, combination of two structures may occur (as illustrated in Figure 3.12). In this case an *interval* (i.e. two notes) is shared between the two structures, thus the process [P] of Figure 3.12 comprises the first four notes, and the intervallic process [IP] comprises the third, fourth and fifth notes (so notes three and four are shared by both structures). As can be seen, there are only two notes (notes one and five) that may be promoted - the initial and terminal notes of the *combination* of the two structures.

Note that the combination of structures is notated by a full stop between the two structure names[18].

---

18 This is our own addition to Narmour's notation — although his notation is unambiguous (since the I's and V's always occur before the P, R, or D of a structure name), it was felt the inclusion of the full stop as a separator makes reading more clear.

3.8.3. Chaining (combination of three or more structures)

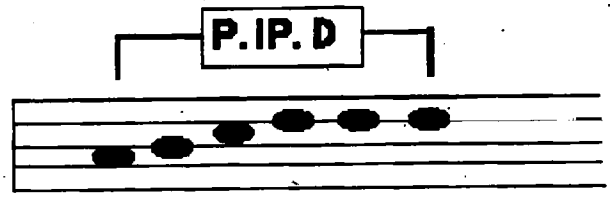

Figure 3.13 : Chaining of three structures.

As mentioned earlier, chaining is the term Narmour give to a situation of three or more structures combining together. As with combination, each pair of adjacent structures shares an interval, and only the initial and terminal notes of the complete chain may be hierarchically promoted. The notation (as illustrated in Figure 3.13) is the same as for combination, except that the list of structure names will be three or more long.

The example in Figure 3.13 above shows a process **[P]** structure for the first four notes, an intervallic process **[IP]** structure for notes three, four and five, and a duplication **[D]** structure for notes four, five and six. In such chains, the same note may play a role in up to three different structures (e.g. note four in Figure 3.13).

3.8. Hierarchical promotion

In most cases the initial and terminal notes of structures (except monads and dyads) are promoted to the next hierarchical level above[19] — exceptions being monads **[M]** (only the one note of a monad is promoted)[20]; dyads **[Interval Size]** are a special case, since the Implication-Realisation Model states that only one of the two notes making up a dyad are transformed (hierarchically promoted)[21]; when combining or chaining **[Dot-list of Structures]** of structures occurs, the initial and terminal notes of the complete combination or chain only are promoted[22].

---

19 Potentially forming a dyad **[Interval Size]** at the next hierarchical level.

20 Potentially forming another monad **[M]** at the next hierarchical level.

21 Again, potentially forming a monad **[M]** at the next hierarchical level.

22 Potentially forming a dyad **[Interval Size]** at the next hierarchical level.

3.8.1. Hierarchical promotion of one note of a dyad

Only one of the two notes making a dyad is promoted (unless the other note is the initial or terminal note of another structure). The four rules describing conditions to determine which note of a dyad should to be promoted are as follows (Narmour 1990, p. 428):

(1) **Stress rule.**

If the dyad is durationally additive and stress (e.g. offbeat accent) is present, then location of stress determines whether the initial or terminal tone is the structurally transformed note.

(2) **Dissonance rule.**

If the terminal note of a dyad in an additive setting is a resolution of dissonance appearing on the initial tone, then the consonant tone functions as the structural note.

(3) **Long-note rule.**

If the dyad is countercumulative[23], then the initial note is structurally transformed; if the dyad is cumulative[24], then the terminal note is (assuming no stress on the short note in the countercumulative patterns or dissonance on the long note in cumulative patterns)[25].

Following from these rules, the number of notes to be promoted from structures at each level are:

- monads promote one note,
- dyads promote one note,
- processive and reversal structures promote two notes,
- chains and combinations of structures promote two notes.

3.9. Bringing together the elements of the theory

Figure 3.14 illustrates the way the different concepts of the *Implication-Realisation Model* combine in a bottom-up fashion, to make up the analytical theory. The sections of this chapter have followed the sequence of concepts illustrated in Figure 3.14 (to be read from the bottom upwards).

---

23 Long note to short note.

24 Short note to long note.

25 Narmour mentions (p. 428) that dyads involving rests are normally evaluated in the same way (e.g. [quaver note, quaver rest, quaver note] is treated as [quaver note, quaver note] etc.)

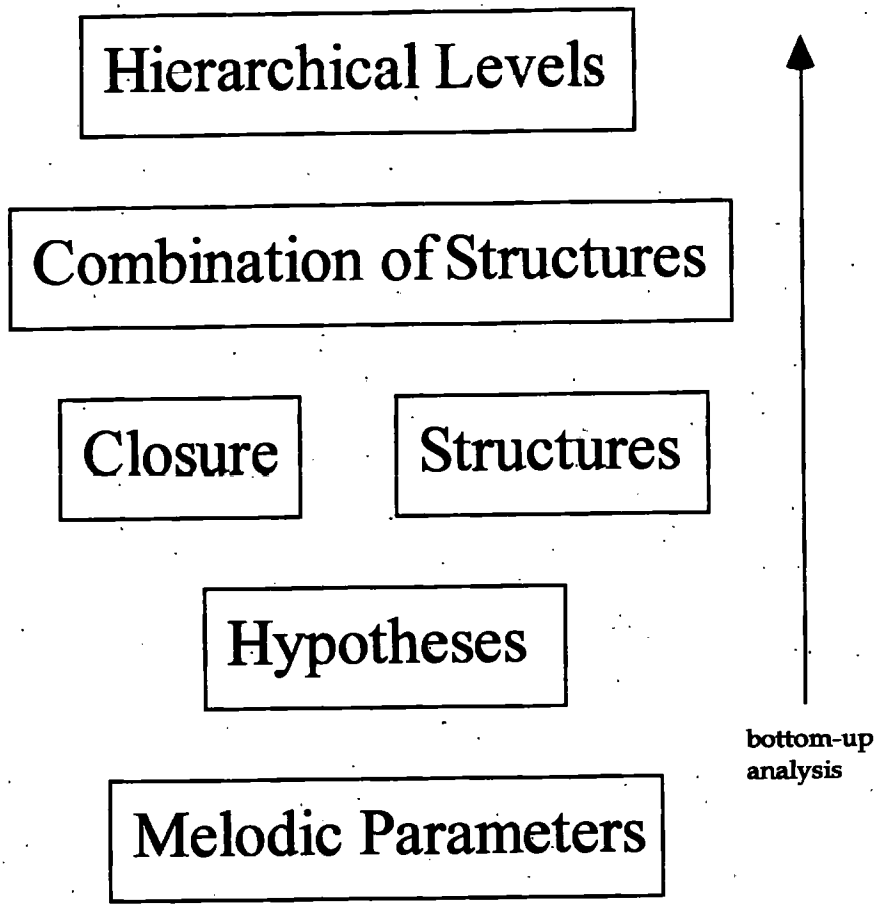

Figure 3.14 : The features of Narmour's Implication Realisation Model.

Figure 3.15 below illustrates the above concepts in a hierarchical analysis of a melody. For parsing of melodies one should work from the bottom of the diagram, upwards.

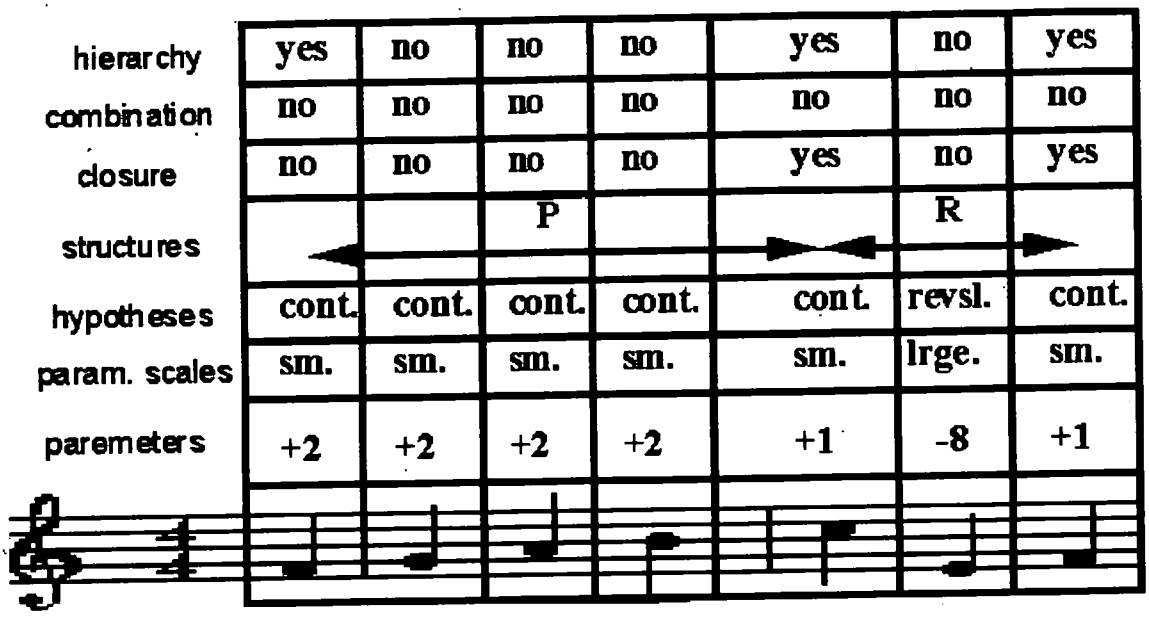

Figure 3.15: Combination of elements of Implication-Realisation model.

3.10. A critique of the Implication-Realisation Model

This section presents a summary of published critiques of Narmour's *Implication-Realisation Model.* This thesis does not attempt to test the psychological validity of the *Implication-Realisation Model,* however, here we summarise published reviews of Narmour's book (1990). These reviews each highlight a number of aspects of the theory, for either criticism, or noteworthiness.

3.10.1. Relation of Narmour's theory to Fodor's model of perceptual processing

Cumming (1992) notes that Narmour's theory is heavily influenced by Fodor's (1983) model of perceptual processing, which proposes a number of separate cognitive systems, some of which act without access to high level expectations. Cumming criticises Narmour's theory with respect to tonality thus:

> *"The weakness of the theory in dealing with the cognitive role of tonality derives, in part, from Narmour's simplification of the model put forward by Fodor. He [Narmour] asserts, in concluding, that 'the theory attempts to meet the condition of a psychological modularity of mind' (p. 425, Narmour 1990), but he nowhere gives an adequate account or ustification of how he divides cognitive modules in auditory cognition."* (p. 370, Cumming 1992)

Cumming goes further in criticising Narmour's model, arguing that while his treatment of parametric

scales separately "*suggests a complex differentiation of perceptual functions concerned with input, each having relative autonomy from the other*" (p. 371), Narmour's treatment of closure confuses the issue (drawing on the interparametric influences of harmony, duration and metre). In addition Cumming suggests clarification is needed regarding the focus of Narmour's theory, asking the question as to whether he is focusing on *the musical parameters themselves* or on the *cognitive modules used in processing parametric information.*

3.10.2. Relation of Narmour's theory to theories of natural language processing and perceptual categorisation

If one is to take a similar parsing approach to melody analysis to the one used for natural language processing a number of issues need to be considered. While Smoliar (1991) concludes that Narmour has taken *"positive action on a new vision of the cognition of music"* (p. 56), he raises four questions about the appropriateness of such a model as Narmour's for explaining the cognition of melody:

- ***Does Melody Have a Syntax?*** Smoliar argues that syntactic analysis for natural language is appropriate, since the units of natural language (words) are symbols (whether spoken or written). However, he argues that the only symbolic structures found in melodies are *"the post hoc ones that show up in our notational systems"* (p. 49). In other words, music analysts wishing to perform syntactic analysis have to regard the notational symbols as the music itself, since music is not made up of symbols in the same way as natural language. If one agrees with Smoliar, the only recourse for syntactic music analysts is to argue that the syntax of the notation directly corresponds to the music being notated - an argument which Smoliar says there is no reason to assume is true.

- **Are Notes the Lexical Primitives of Melodic Syntax?** Given that melody does have a syntax, should notes be the lexical primitives one uses for syntactic analysis? nce again, Smoliar agrees that *"if one equates melody with the way it is notated"* (p. 49) then notes are the primitives of melody.

- **Perceptual categorization: An Alternative Perspective** Perhaps the strongest potential criticism of Narmour's theory is Smoliar's discussion of experimental evidence from Edelman (1987) of what were thought to be "hard wired" perceptual archetypes. Smoliar states that Edelman's research has concluded that:

  "these archetypes ... were not universal. Instead, they are products of early sensory experience. For example, an animal deprived of any exposure to vertical lines will not develop the necessary archetypes for recognising them". (Edelman 1987, p. 51)

  Smoliar discusses Edelman's work further, stating that in addition to the development (or not) of low level perceptual archetypes, the same mechanisms drive the development of the higher level cortical organisation that interprets the signals from the archetypes (i.e. by sensory experience). Such research would seem to argue against the kind of perceptual universal primitives proposed by the *Implication-Realisation Model*, i.e. Narmour's low level perceptual modules are not "hardwired" but develop, and so will develop differently between individuals exposed to different auditory stimuli (i.e. music cultures and styles) when young. However, Edelman's work is based on the perception of static visual images, therefore (as Smoliar points out), until such research accounts for the perception over time that auditory perception involves, the consequences of Edelman's research for Narmour's theory are unclear.

Another criticism raised by Smoliar, is that while theories of natural language must deal with ambiguity, Narmour's theory does not. He states:

> *"The idea that it may be possible to 'read' a melody in such different ways, all of which may be relevant to understanding is apparently not part of Narmour's view of melodic syntax. Associated with any melody is a single analysis that is basically an absolute truth".*
>
> (Smoliar 1991, p. 45)

A defence of Narmour's theory to such a criticism is that there are two aspects to his theory — the *bottom-up*, "built-in" low level cognitive processes, and the *top-down* influence of style. While the bottom-up aspects of his theory yield only a single hierarchical analysis of a given melody, Narmour's statements about the effects of top-down style suggest that many different parses of the same melody are possible, according to which style structures one takes into account for the listener being modelled. The research in this thesis is based on solely the bottom-up aspects of the

Implication-Realisation Model. Although Narmour refers to top-down stylistic influence in many of his analyses, it is only the bottom-up aspects of his theory that have been published in sufficient detail for computational modelling. Later in this thesis extensions to our work are described, taking account of how formalised descriptions of the influence of top-down structure could be utilised for both more sophisticated analysis of melodies and more effective aid for novice composers (see the further work section of Chapter 8 describing the components of our proposed intelligent learning environment, and also describing extensions and further work leading from the research described in this thesis).

3.10.3. Relation of Narmour's music theory and the psychology of music

Butler (1992) raises a number of points of criticism in his review of Narmour's book:

- Narmour states that monads generate no implications, but Butler argues that, though there may be no bottom-up implications generated from a monad, such attributes of the tone such as duration, volume, pitch and timbre may well trigger a top-down style structure, thus expectations may be generated. (p. 244)

- The inclusion of registral return as an archetype (structure) is questioned, since it describes relations between non-contiguous melodic notes, and is non-implicative. (p. 246)

- There is no psychological basis for the reversal hypothesis — but Butler notes that Narmour is sure such a general principle of perception will be identified. Butler also proposes that the reason why no reversal principle can be found mentioned in the psychology literature is that *"the lion's share of the literature of Gestalt psychology - and of psychology as a whole - has had to do with vision, and Gestalt principles have dealt primarily, almost exclusively, with the perception of static visual arrays"*. (p. 251)

- It is perhaps interesting to note that in a footnote, Butler puts forward a view that is almost the reverse of the findings of Edelman (1987) and the concept of perceptual categorisation. Butler suggests that in addition to the "hard wired" low level implications being driven by Gestalt laws of perception, similar laws may have influenced the composers — he writes:

  "If it is true that Gestalt principles somehow describe the musical listener's expectations today, why wouldn't it also be true that these same principles (though then unnamed) somehow describe some of the conscious and unconscious choices of the composers who crafted the music from which those style traits derive?" (p. 248)

In his review, Butler briefly discusses some aspects of the process of analysing a melody in terms of the *Implication-Realisation Model*. He makes the following statement (the example to which he refers is reproduced in this Chapter as Figure 3.16):

> *"An analyst would first identify metrical groupings and patterns of durational cumulation (i.e. short note to long note) and consider the notes that make up each grouping, bracketing the generative interval and one or more intervals of realization. Groupings could be separate or elided; in this example, groupings dovetail at metrical boundary points at two different levels."* (Butler 1992, p. 247)

Both the statement of analytical process, and the analysis itself are noteworthy. Narmour's *Implication-Realisation Model* describes how a listener processes the notes of a melody as they are heard, note-by-note in chronological sequence. However, Butler is stating above that before considering the intervals of the notes of the melody an analyst should identify metrical groupings and durational cumulations. This kind of holistic approach to melodic analysis does not fit with the implications generated note-by-note as described in the *Implication-Realisation Model*. This mismatch of theory and the process of its application highlights the fact that Narmour does not attempt to describe a theory of metric analysis (nor harmony for that matter), that also works on a melody in a note-by-note fashion (such as Longuet-Higgins and Lee (1982, 1984), Lee (1984)).

3.10.4. Ambiguities in Narmour's theory, noted by critics

Butler discusses the analysis shown in Figure 3.16. This analysis is actually at **three** levels: the retrospective reversal **[(VP)]** and process **[P]** of the last five notes (level 1); the three intervallic

process [IP] structures at level 2 — note that only the promoted notes (B, D and E) of level 1 make up the third [IP]; level 3 is the single process [P] formed by the promoted initial and terminal notes of the three intervallic processes of level 2. What neither Butler, nor Narmour, explain is under what conditions notes "skip" a level for analysis — i.e. why are only the last five notes of Figure 3.16 considered for level 1? Narmour does mention that the theory is not always rigidly applied, but does not clarify the conditions for when it is not to be so.

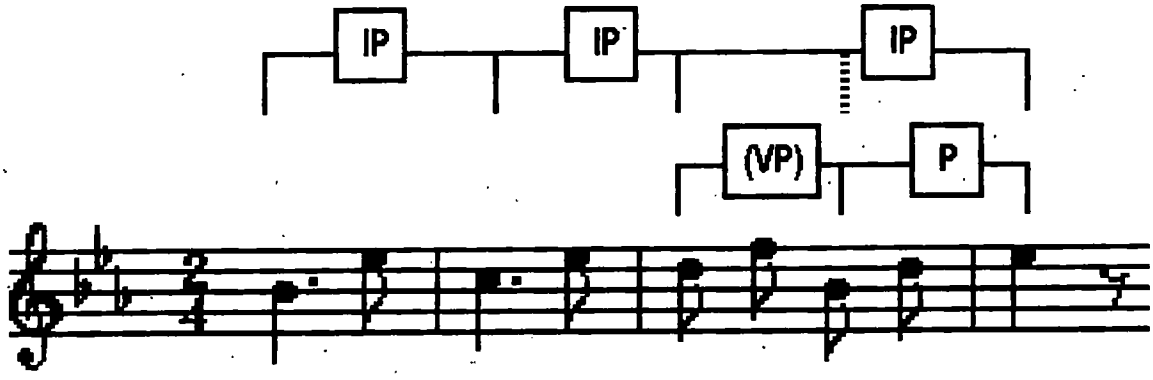

Figure 3.16: Butler's (1992, p.249) analysis of the primary theme from the third movement of the Haydn concerto for Trumpet and Orchestra.

An example of the ambiguity of Narmour's description of the theory is illustrated in Butler's compact table describing the criteria for the different structures of the theory (Butler 1992, Table 2, p. 247). In the table he states that both intervals of a prospective process [P] structure are an augmented fourth or smaller, however Narmour states explicitly:

> *"Concerning such patterns that mix large and small intervals, there is, however, one methodological detail to be noted for the sake of a consistent application of the symbols: if the first interval of a process mixing sizes of intervals begins with a reversal interval (P5 or larger), then I shall analyze the whole pattern as a retrospective process [(P)]. If the first interval is a continuation interval (P4 or smaller), then I shall analyze the pattern as a prospective process [P]. ...*
>
> *With reference to processes that begin with tritones or diminished fifths, usually harmonic context makes it clear whether the whole pattern creates a retrospective or a prospective realization."* (Narmour 1990, p. 274)

As can be understood from the above quotation, when dealing with the threshold intervals of a perfect fourth up to a perfect fifth, the category of structure by which a given fragment of melody is analysed

depends on both harmonic context and the order of threshold intervals. Thus the Implication-Realisation Model defies simple definition.

3.10.5. Relation of Narmour's theory to harmony

> *"... Narmour considers melody independently from harmony and hardly allots harmony a role in the delineation of melodic structure. It comes as no surprise that, given his conception of structure, Narmour often designates dissonant pitches as structural. ... Conversely, when resolutions of dissonance are only fleetingly presented (...), he does not regard those resolutions as structural notes."* (Krebs 1989, p. 434)

Krebs criticises Narmour for reducing the influence accorded to harmony in the structuring of melodic analysis. As described earlier in this chapter, the of notes duration plays an important role in determining whether a note is structural or not (i.e. whether it has strong closure, and so is an initial or terminal note of a process) — Krebs argues (as illustrated in the quotation above) that such an emphasis means that at times (long) dissonant notes are structural while (shorter) notes of harmonic resolution are not.

This issue is clearly one of whether Narmour's model can effectively model the top-down influence of harmony in the calculation of closure (and thus structural importance) of notes. Krebs states that his disagreement with Narmour is due to the separate analysis of different parameters:

> *"I cannot conceive of melody and harmony as independent parameters in tonal music. ... Since I cannot, like Narmour, divorce my hearing of tonal melodies from harmony, virtually all of his analyses 'go against the grain' as far as I am concerned"*
>
> (Krebs 1989, p. 435)

Psychological experimentation may lead to answers as to whether humans do separately process the parameters of music, and whether the parameters Narmour has based his theory upon are the rights ones. Although some music analysts may disagree with the results of Narmour's theory, they are able to criticise his analyses in terms of the theory — thus, even if the theory has flaws by being an (informal) attempt to approach the task of music analysis from the direction of cognitive psychology, Narmour offers much more than a collection of contentious analyses.

3.11. Psychological validity and strengths of the theory

> *"At the most fundamental level, the implication-realisation model stands apart from virtually all other musical theories because it is based ultimately on psychological principles rather than on appeals to acoustics, mathematics, or compositional convention."*
>
> (Butler 1992, p. 249)

There is preliminary support for the *Implication-Realisation Model* from empirical psychological experiments (for example Krumhansl 1991; Krumhansl & Shellenberg 1990; Shellenberg & Krumhansl 1991). In addition, there is some support for the use of a Gestalt psychology approach for the modelling of music perception; the remainder of this section summarises Narmour's discussion on the use of Gestalt psychology.

The use of Gestalt Psychology for a model of music perception

It may seem unusual for a bottom-up theory of melody analysis to be based on what have been considered by some as a purely top-down and outdated theory of perception — for example some traditionalists claim Gestalt laws are inherently top-down (e.g. Bower & Hildgard 1981). Others disagree and argue that Gestaltism contains a mixture of top-down and bottom-up aspects (e.g. Pomerantz 1981, p. 163, cited by Narmour, p, 63). Criticisms of Gestaltism (Marr, 1982, p. 196, cited by Narmour 1990, p. 69) highlight the way the theory maintains that primitives are meaningless, then goes on to include inference rules based on the combination of primitives (such as the law of similarity). Marr talks of how Gestaltists failed to "*appreciate the complexity of functions that can be computed by local interactions*" (Marr, p. 196). Narmour cites Deutsch (1982a, 1982b, cited by Narmour 1990, p. 63) as additional defense of the relevance of Gestalt laws for the study of the cognitive psychology of music. Narmour attempts to follow the work suggested by Meyer (1956) in applying bottom-up aspects of Gestaltism in the formulation of a cognitive theory of melody analysis. In fact the bottom-up analyses of the Implication-Realisation Model could be seen as an attempt to take such an approach to an extreme, whereby the entire analysis is based on Gestalt concepts such as similarity and proximity.

3.12. Conclusions on presentation of Narmour's theory

It appears that the *Implication-Realisation Model* is a unique model of music analysis, in that it is based on psychological principles, applied to low level parameters. Two prominent questions about Narmour's theory that remain open relate to the empirical testing of the theory for psychological validity, and one of a more theoretical nature:

- whether the psychological theories are correct (hence Narmour's list of experiments[26]), and
- whether Narmour's theory is a competent application of the psychological theories in question (this is Cumming's (1992) main criticism — that Narmour's theory is limited due to its simplistic application of Fodor's (1983) model of perception).

Narmour's theory, of all those reviewed in Chapter 2, is the only one that is constructed from the bottom-up on the basis of theories of music cognition. The Implication-Realisation Model is potentially more general, due to its intrinsically melodic rather than harmonic basis. Narmour's theory has been published for a number of years now, and despite there being a number of open questions that remain about the theory it appears to have resisted criticisms such as have been described in this review. In a recent book by Robert Rowe (1993), the three analytical theories discussed were Schenkerian Analysis (1956), Lerdahl and Jackendoff's generative theory (1983), and Narmour's *Implication-Realisation Model* — this would appear to be a fair placement of the importance of Narmour's research.

Narmour presents his theory in much detail, both through the explicit methods such as a description of his rules of influence and how they work for the different parameters of melody, and through his presentation of the theory via numerous musical examples. Although there are one or two weaknesses in his presentation of the theory (such as the procedural steps to follow in a parse, and some definitions of closure — these issues are addressed in the next chapter), Narmour's theory is both in a form that is suitable for computational instantiation, and described sufficiently for the process of creating a formal model for implementation. The theory is not limited to any particular style of music, and Narmour has proposed ways that the mode might be extended for non-Western Tonal

---

[26] When discussing the importantce of pscyhologically testing theories of music cognition, Narmour (1990, pp. 418-423) lists twenty-one experiments to test aspects of the validity of psychology foundations of the *Implication-Realisation Model.*

Musics. Since the computer model of the theory is intended to be used as part of a system to aid novice composers, one appealing aspect of the model is that, due to the bottom-up basis for parsing, it works well with fragments of melodies. The combination of these various strengths of the theory itself, and the form and detail of Narmour's presentation of the theory, result in a strong argument for the use of the *Implication-Realisation Model* as a basis for a computational system for melody analysis (and generation).

Chapter 4:
Formalisation of Narmour's Theory

4.1. Aims and need for a formalisation process

This chapter presents a clarification[1] and formalisation of Narmour's *Implication-Realisation Model*. This modified version of the theory is the basis of a declarative, logic-based parser (described in Chapter 5). The process of clarification involved critical review of Narmour's publications (Narmour, 1984, 1989, 1990 and 1991)[2]. The aim of this work has been to present an unambiguous and explicit statement of the theory, from which to develop a computational model. During the clarification process the theory has been modified, and in some cases simplifying assumptions have been made. Where such assumptions have been made they are identified. An advantage of our formal and computational implementation is the opportunity the model offers to fine tune assumptions behind the modified theory — analysts may wish to use different definitions for concepts of the theory (perhaps for different musical genres or tonal systems, or more sophisticated definitions if an analysis is to focus on the influence of, say, metre, on melody). In addition changes could be used to explore alternative analyses of a single melody.

---

[1] The clarification process has involve *disambiguating* Narmour's descriptions of his theory, stating *choices* when alternative interpretations were possible, and making *simplifing assumptions* about unspecified top-down aspects of the parsing process (such as how to measure of metric strength for closure).

Specifically, the clarification and formalisation process had the following goals:

- to locate any gaps in the theory,
- to identify ambiguities, to describe the alternative interpretations and justify decisions we have made during the formalisation process,
- to quantify systematically Narmour's statements and theory definitions[3],
- to check the theory for consistency, and
- to use a predicate-calculus formalism where appropriate to make concrete imprecise verbal descriptions of the theory.

The formalisation process has been necessary due to the informal description of the *Implication-Realisation Model* (e.g. Narmour 1989, 1990, and 1991), both in terms of *content* and *style*. The shortcomings of the content are considered in the rest of this chapter. The style of Narmour's existing descriptions of his theory also presents a hurdle to its unambiguous description, simply because at times it is unclear what is being described in some statements, and how specific the application of the statements is. Smoliar (1991) has a number of criticisms about Narmour's description of the theory:

> *"Unfortunately, style can often inhibit communication just as powerfully as it can facilitate it; and style is probably the greatest liability in this book."*
>
> *"Sometimes one needs the gusts of a fierce hot wind to blow away the cobwebs of a stagnant discipline. ... Because Narmour never seems to resist the opportunity to let loose another gust, the reader is often hard pressed to get at the story he is trying to tell."*
>
> *"... we are exposed to impressions of results in psychology, physiology, philosophy, mathematics, artificial intelligence, and cognitive science ... However, many of these impressions can be misleading, if not downright inaccurate."*
>
> *"Sometimes Narmour is just sloppy in his use of terminology."*
>
> (Smoliar 1991, pp.53-54, reviewing Narmour 1990)

---

2 The most detailed and complete description of the *Implication-Realisation Model* is in Narmour's book (Narmour, 1990) — it is on this presentation of the theory that the greater part of this, and the previous, chapter is based.

3 For example we have introduced numerical representations for overall and individual measures of closure — described later in this chapter.

Some of the other reviews of Narmour's theory have similar, if less outspoken, criticisms of Narmour's descriptions, for example Cumming's (p.372, 1992) *"This book makes quite stringent demands on the reader's capacity to suspend reference to any previously learned theory, objectify herself from her own musical experience and assimilate a new terminology. In simple terms, it is not an easy read"*. Other researchers may disagree with the simplifications presented in this chapter (and choices made in cases of ambiguity), however the modified theory presented here describes just one set of choices; a more important aim of the work in this chapter (and Chapter 5) is to provide a clear presentation of our clarification of Narmour's theory, to make such criticisms, and adoption of different choices, possible.

Although Narmour's description of his theory is lengthy, complex and at times poorly expressed, this is perhaps inevitable with any theory presented in as much detail as the *Implication-Realisation Model*. It is from the numerous examples that we have been able to infer aspects of the theory not stated explicitly by Narmour (such as the chronological procedure, at multiple levels, for the parsing of melodies, described in the next chapter).

The remainder of this chapter is presented in three stages:

- definitions of closure,
- the concept of regular, recursive hierarchical application of the Implication-Realisation Model, and
- the formalisation of the definitions of structures by the use of predicate-calculus notation.

4.2. Definitions of closure

Most of the definitions of closure given by Narmour (for example due to stopping, or cumulative duration) are precise and unambiguous. However, definitions of closure due to metre, harmony, top-down style, and the method of quantifying the effects of a combination of closure are not fully defined in his presentation. Each of these aspects of closure are considered, and formal definitions made in below.

4.2.1. Metric closure

Metric closure is the closure on a note due to the importance (or "strength") of the metric "pulse" at the onset time for the note in question. Narmour provides no definition of metric

strength in his descriptions of the *Implication-Realisation Model*. For the purposes of presenting a complete and quantified version of the *Implication-Realisation Model* we here present a measure of metric closure based on Levitt's (1985) hierarchical representation of metre. Levitt's representation has been chosen for its simplicity, its explicit, quantified form, and the fact that it is hierarchical (which provides a good basis for the application of metric constraints on levels in a hierarchy for the generation of melodies, explored in Chapter 7).

The metric strength of a given note, according to Levitt (and computationally modelled in a constraint-based model for melody generation, Smith, 1990), can be defined as a function of the smallest common denominator of the onset time of a note, with respect to the set of associated denominators calculable from the metre within which the note occurs. Levitt describes metres in terms of a list of factors (called by Holland (1989) a *metric vector*[4]).

For example music with a metre of 4/4 time would have a list of factors of: [1, 2, 2, 2]. This represents that important onset times for notes in this metre are:

[1, ...]	at the onset of a bar,
[1, 2, ...]	onset of a bar divided by 2 (i.e. middle of a bar),
[1, 2, 2, ...]	middle of a bar divided by 2 (i.e. quarter bar or three-quarter bar onset, and
[1, 2, 2, 2]	quarter of a bar divided by 2 (i.e. onset of an eighth bar, three eight's, five eight's or seven eight's of a bar).

By multiplying these factors, we get the following list of denominators (for use in function described below):

[1, 2 (1 * 2), 4 (1 * 2 * 2), 8 (1 * 2 * 2 * 2)]

This is a list of the denominators representing how metrically important a note is by dividing the onset time of a note (from the beginning of the bar) by the total duration of the bar. For the given denominators above, the most metrically important note are those whose onset is the

[4] See Appendices C and D for a description of the representation of melodies and the calculation of the metric strength of notes based on metric vectors.

same as the beginning of a bar (denominator = 1). The next most metrically important note in a bar is one whose onset time is half the duration of the bar (denominator = 2). The next most metrically important notes are those whose onsets are on the quarter bar and three-quarter bar time points (denominator = 4).

An example of the formalised definition of metric closure

Figure 4.1 below shows a fragment of a melody (with its *Implication-Realisation Model* analysis, reproduced from Narmour 1989, Figure 8, p. 54: Bach, *Mass in B Minor,* Kyrie Eleison II, mm.1-3). In this example all promoted notes have relatively strong closure due to metre and duration. We can illustrate how the use of metric vectors identified the closure due to metre by analysing the simplified[5] representation of notes and their onset times in listed in Figure 4.2. Also in Figure 4.2 is the representation of the metric vector associated with the melody fragment in question. The onset times are expressed using a simple discrete representation of time (MOTIVE time units, described in Appendix C).

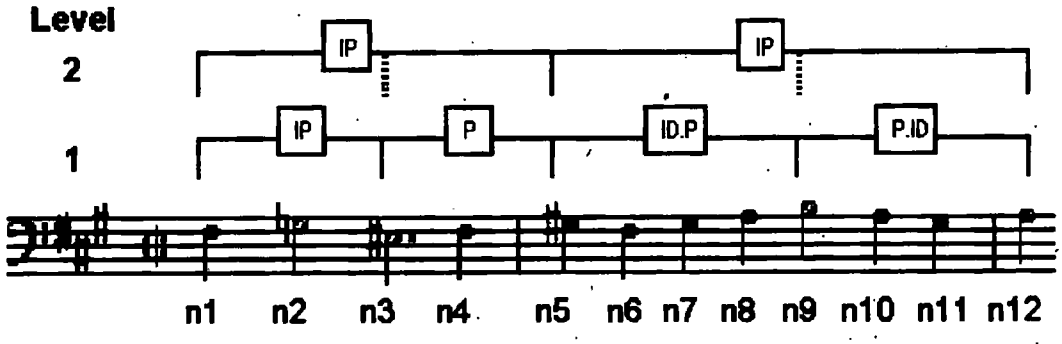

Figure 4.1: A melody and its graphical analysis.

In the simplified representation in Figure 4.2 below, the "metre_list" predicate states that a metre starts with onset time of zero, and has a metric vector of [1, 2, 2, 2]. For this example the total duration of a bar is 529200 time units.

[5] A section in the follow chapter gives a detailed description of the full representation of melodies and analyses used by our parser.

```
metre_list([[   0 , 1,2,2,2], ... ]).
note( 1, 0 ).
note( 2, 132300 ).
note( 3, 264600 ).
note( 4, 463050 ).
```

Figure 4.2: Simplified representation of the metre and onsets for the first four notes of the melody from Figure 4.1.

The simple algorithm for identifying the metric strength can be expressed informally as follows:

1. If onset = 0, treat it as if it were the total duration of the bar)
2. simplify the rational of (note onset / total bar duration) as much as possible
3. find earliest denominator of from the list generated from the metric vector

Applying this function to each of the onsets results in the following:

note 1, onset = 0

treat as bar duration (529200),

divide by bar duration and simplify = 529200 / 529200 = 1,

matches 1st denominator

note 2, onset = 132300,

divide by bar duration and simplify = 132300 / 529200 = 1 / 4,

matches 3rd denominator

note 3, onset = 264600,

divide by bar duration and simplify = 264600/ 529200 = 1 / 2,

matches 2nd denominator

note 4, onset = 463050,

divide by bar duration and simplify = 463050 / 529200 = 7 / 8,

matches 4th denominator

The measure of metric strength of each note is equal to the position of the denominator of the simplified rational (the closer to the beginning of the list of denominators, the stronger the metric strength). For the notes above, note 1 has most metric strength, then note 3, then note 2, and finally note 4. In the analysis closure and promotion occur on notes 1 and 3.

4.2.2. Harmonic closure

Harmonic closure is prospective (i.e. occurs on a note, rather than due to a following note or

rest), and occurs when a note is more dissonant that the preceding note. Narmour provides no definitions for how to measure the consonance of notes.

We have adopted for our clarification of Narmour's theory the use of a measure of consonance based on a set of *Pitch Spaces* (Lerdahl, 1988). These are described in detail in Chapter 7, where the same approach is used for the application of harmonic constraints for notes in a melody to be generated. A review of Lerdahl's theory has been presented in Chapter 2.

In brief, the pitch spaces define a set of hierarchical levels, each higher level describing pitches that are more consonant than those below (consonance is determined by reference to the key (or mode) and chord associated with the note in question). The more difference between pitch spaces that two notes exhibit, the greater the harmonic difference, and if the difference is from a consonant note to a dissonant one, harmonic closure is said to have occurred. This measure of harmonic closure is much simpler than would be applied by a human music analyst, but is sufficient for our purposes in that it is both unambiguous, and suitable for computational implementation, and educational use by novice composers of melody.

4.2.3. Stylistic closure

Closure due to the top down influence of style is discussed generally by Narmour, and is indicated in many of his analyses, but never described in any kind of detail — neither the cognitive mechanisms for recognition of stylistic closure when analysing a melody, nor the mechanisms for the learning of new style structures when listening to melodies. The definitions and mechanisms for stylistic closure have not been modelled in our formalised theory, and do not form part of the computational parser; the requirements of such work are outlined as an independent project in the further research section of the final chapter of this thesis.

4.2.4. Simultaneous occurrence of different forms of closure

The above forms of closure, although described in some detail by Narmour, still do not fully define what strength of closure occurs on a note for any given melody — in Narmour's descriptions there is a lack of any kind of definition for metric strength, harmonic dissonance or how style is learnt, represented or applied by a listener. The situation is further complicated when two or more forms of closure act upon a single note in a melody. This raises the question of how to determine the combined closure strength on a note.

We have done this by a two stage approach: first, each form of closure on a note is analysed, and an integer value calculated (representing the strength of each type of closure); second, a total closure calculation is made, where the numeric strength for each type of closure is multiplied by a weighting factor, and the total of these weighted closure is the sum closure on a note. The numbers and weightings used are discussed in the next chapter, as part of the description of the computational model of the clarified theory. This additive method of combining closure is a partial simplification, since in some cases (for example when dissonance and durational closure, see Narmour 1990, p. 108) when two forms of closure occur on the same note, the result is actually less, rather than more, closure.

4.2.5. Combination of structures due to weak closure

When (due to weak closure) two or more structures share intervals, a contiguous sequence of structures occurs, for which only the first note of the first structure, and the last note of the last structure in the sequence may be hierarchically promoted (hierarchy is discussed later in this chapter) — for an example of such a sequence see Figure 3.11 in the previous chapter. Although this is a single concept, Narmour distinguishes between cases where only two structures share an interval, which he calls combining, and cases where three or more contiguous structures share intervals, which he calls *chaining*.

4.2.6. Definition of degree of closure strength

In our modified version of Narmour's theory, we classify closure strength as one of four forms.

Separate

Closure of sufficient strength on a note to cause the note to be promoted, and to terminate the current structure such that no notes are shared with the next structure for the level.

Shared

Closure of enough strength to cause the note to be promoted and to terminate the current structure, but weak enough to cause the note to become the first note of the next structure for the level (thus the note is shared between two structures, being the last of one, and the first of the following structure).

Merge

Very weak closure, sufficient to terminate the current structures implications, but insufficient

to completely close the structure — the result is that a chain (or combination) of structures occurs, with the last interval of the current structure being shared with the next structure for the level (i.e. the note with "merge" closure and the previous note are shared between two structures within a combination or chain).

Negligible

No closure, or extremely weak closure, resulting in no blunting of the current structure's implications sufficient to cause any kind of structure termination.

It is unclear from Narmour's informal descriptions of his theory whether our definitions of separate, share and merge closure directly correspond to Narmour's classification of transformational, formational and articulative closure respectively.

The three closure strengths greater than "neglibible" result in three different groupings of notes in structures. Each of these is described in the following sub-sections. A fourth sub-section is included, to clarify how Narmour distinguishes between the merging of two structures ("combination"), and the merging of three or more structures ("chaining").

Chain / merge

There appears to be no good reason why Narmour distinguishes between merging of two structures (combination) and merging of three or more (chaining).

4.3. A problem with hierarchical promotion

Although in most of Narmour's analyses the notes forming structures at one hierarchical level are all notes from the level immediately below, on some occasions Narmour allows notes to "skip" a level. Narmour briefly mentions that his theory does not always systematically apply recursively:

> *"One must not think, however, that such recursion is always systematic. For music is not a systematic phenomenon but rather a truly hierarchical one (Narmour 1983). Since true hierarchies are inherently discontinuous (as opposed to systemic), one cannot invoke the production sequence mechanically. For context can short-circuit the normal progression of events. Within a context of dissonance, for instance, it is possible for the first structural tone of, say, a process to be articulative (ar) or*

> *formational (fm) — and thus remain on the low level — while the second structural tone alone becomes transformational. Thus hierarchical production would proceed directly from process [P] to monad [M], omitting the dyad."*
>
> (Narmour 1990, p. 413)

In this clarification of Narmour's theory we do not allow non-systematic application of hierarchical promotion. However, in cases where Narmour would have skipped a level our formalised theory can reproduced the promotion of notes to higher levels by the use of monads (see Chapter 6 for details). Unfortunately, Narmour has not described under what conditions such skipping of levels occurs, therefore todate we have had to hand code each melody with the monad closure to get a duplication of Narmour's higer levels of analysis involving notes from lower levels.

A second problem with the concept of recursively applying the Implication-Realisation Model parsing techniques to melodies at higher levels is the question of what fills in the gap for non-promoted notes (e.g. if a melody contained a sequence of notes N1, N2 and N3, and N2 was not promoted, what happens to the gap at the higher level?). Two solutions suggested themselves, the first is simply the insertion of rests for each non-promoted note, the second is the extension of the duration of the preceding note (e.g. N1 would also gain the duration for N2). In our formalised version of Narmour's theory we have opted for the second of these alternatives, but the metric strengths of notes are not recalculated at the higher level, so each note when promoted always retains the metric strength it was awarded for its onset time in the musical surface.

4.4. Predicate-calculus definitions of stuctures

This section presents a formalised version of Narmour's statements defining the properties of note intervals that determine which "structure" a given sub-sequence of notes belongs to. This section refers directly to the corresponding section in the previous chapter (section 3.6), and for brevity only includes a restatement of our summary of Narmour's statements and our predicate calculus formalisation.

The following symbolic conventions are used in the predicate calculus statements:

- "I" represents an interval;

- "P4", "m3" etc. the letter/step notation is used to represent intervals (the two examples are a perfect forth and minor third);
- "process", "duplication" etc. is the set of all intervals for the structure class being defined;
- "abs(n)" is a function to return the positive value of an integer "n";
- "contour(I)" is a function returning "ascent", "lateral" or "descent" corresponding to the signed integer representing the interval it takes as a parameter;
- the subscription indicates the position of intervals in a given structure, the first interval of a structure is notated I_0 (being the interval from the first to the second note of the structure), the second I_1 (being the interval from the second to the third note of the structure) etc.

Formal definition of process: P

(1) All intervals must be a perfect fourth or less.

(2) The difference between any two contiguous intervals in the process must be no larger than a minor third.

(3) The registral direction does not change.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (I_n \leq P4) \wedge & \\
(2) & (abs(I_{n-1} - I_n) \leq m3) \wedge & \\
(3) & (\ contour(I_n) = contour(\ I_0)\ ) & \rightarrow (I_{0..n} \in \text{ process})\ ]
\end{array}
$$

Figure 4.3: Predicate calculus definition of process.

Formal definition of intervallic process: IP

(1) All intervals must be a perfect fourth or less.

(2) The difference between any two contiguous intervals is a major second or below.

(3) The registral direction of the second interval is different to that of the first.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (I \leq P4) \wedge & \\
(2) & ((abs(I_{n-1} - I_n) \leq M2)) & \\
(3) & (contour(I_1) \neq contour(I_0)) & \rightarrow (I_{0..n} \in int\_process)\ ]
\end{array}
$$

Figure 4.4: Predicate calculus definition of intervallic process.

Formal definition of registral process: VR

(1) All intervals must be a perfect fourth or less.

(2) Registral direction does not change.

(3) There is an interval that differs by more than a major third from its predecessor.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ (\exists\ m)\ [& (I \leq P4) \wedge & \\
(2) & (contour(I_n) = contour(I_0)) \wedge & \\
(3) & (abs(I_m - I_{m+1}) > M3) \wedge (m < n) & \rightarrow (I_{0..n} \in reg\_process)\ ]
\end{array}
$$

Figure 4.5: Predicate calculus definition of registral process.

Formal definition of duplication: D

(1) A pitch is repeated.

(2) A duplication structure must contain at least three notes (three notes are required to form the first two intervals; n=0 & n=1).

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (I = 0) \wedge & \\
(2) & (n > 0) & \rightarrow (I_{0..n} \in duplication)\ ]
\end{array}
$$

Figure 4.6: Predicate calculus definition of duplication.

Formal definition of intervallic duplication: ID

(1) All intervals must be a perfect fourth or less.

(2) The difference between any two contiguous intervals is a minor third or less.

(3) The contour of the second interval is different to that of the first.

(4) The magnitude of each interval is the same.

(1) $(\forall I)$ [	$(I \leq P4) \wedge$	
(2)	$(abs(I_n - I_{n+1}) \leq m3) \wedge$	
(3)	$(contour(I_1) \neq contour(I_0))$	
(4)	$(abs(I_{n-1}) = abs(I_n))$	$\rightarrow (I_{0..n} \in int\_duplication)$]

Figure 4.7: Predicate calculus definition of intervallic duplication.

Formal definition of reversal: R

A *reversal* structure **[R]** is a sequence of three notes: $(I_{0..1})$[6].

(1) The first interval is large (a perfect fifth or more).

(2) The interval between the second two notes is smaller by at least a minor third.

(3) Registral direction changes.

(1) $(\forall I)$ [	$(I_0 \geq P5) \wedge$	
(2)	$((I_0 > I_1) \wedge (abs(I_0 - I_1) > m3)) \wedge$	
(3)	$(contour(I_1) \neq contour(I_0))$	$\rightarrow (I_{0..1} \in reversal)$]

Figure 4.8: Predicate calculus definition of reversal.

Formal definition of intervallic reversal: IR

(1) The first interval must be a perfect fifth or larger.

(2) The second interval must be smaller, and differentiated from the first by a major third or more.

(3) Registral direction is the same.

---

[6] All reversal (whether full, interallic, registral, prospective or retrospective) structures consist of three notes only due to inherent closure. This is explicit in the subscription of the formal definitions for all these classes of structre, since two contiguous intervals only containt three notes.

(1) $(\forall I)$ [	$(I_0 \geq P5) \wedge$	
(2)	$((I_0 > I_1) \wedge (abs(I_0 - I_1) > M3)) \wedge$	
(3)	$(contour(I_1) = contour(I_0))$	$\rightarrow (I_{0..1} \in int\_reversal)$]

Figure 4.9: Predicate calculus definition of intervallic reversal.

Formal definition of registral reversal: VR

(1) The first interval must be either a diminished fifth, or an interval of a minor sixth or more.

(2) The second interval must be larger than the first.

(3) Registral direction changes.

(1) $(\forall I)$ [	$((I_0 = d5) \vee (I_0 \geq m6)) \wedge$	
(2)	$(I_0 < I_1) \wedge$	
(3)	$(contour(I_1) \neq contour(I_0))$	$\rightarrow (I_{0..1} \in reg\_reversal)$]

Figure 4.10: Predicate calculus definition of registral reversal.

Formal definition of retrospective duplication: (D)

As for prospective duplication. See discussion in final section of this chapter why we have not included a definition for this structure.

Formal definition of retrospective intervallic duplication: (ID)

(1) All intervals in such a process need to be a perfect fifth or larger (except the first which may be a diminished fifth).

(2) All intervals in the sequence are the same.

(3) The second contour is different from the first.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (\ (I_n \geq P5) \vee (I_0 = d5)\ ) \wedge & \\
(2) & (\ abs(I_0) = abs(I_1)\ ) \wedge & \\
(3) & (\ contour(I_1) \neq contour(\ I_0)\ ) & \rightarrow (I_{0..n} \in retro\_int\_duplication)\ ]
\end{array}
$$

Figure 4.11: Predicate calculus definition of retrospective intervallic duplication.

Formal definition of retrospective process: (P)

(1) All intervals in are a perfect fifth or larger.

(2) All contiguous intervals in the sequence differ by a minor third or less.

(3) All intervals have the same registral direction.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (\ I_n \geq P5\ ) \wedge & \\
(2) & (\ abs(I_{n-1} - I_n) \leq m3\ ) \wedge & \\
(3) & (\ contour(I_n) = contour(\ I_0)\ ) & \rightarrow (I_{0..n} \in retro\_process)\ ]
\end{array}
$$

Figure 4.12: Predicate calculus definition of retrospective process.

Formal definition of retrospective intervallic process: (IP)

(1) All intervals in such a process need to be a perfect fifth or larger (except the first which may be a diminished fifth).

(2) All intervals in the sequence are less than a major second.

(3) The contour of second interval is different from the first.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (\ (I_n \geq P5) \vee (I_0 = d5)\ ) \wedge & \\
(2) & (\ abs(I_0 - I_1) < M2\ ) \wedge & \\
(3) & (\ contour(I_1) \neq contour(\ I_0)\ ) & \rightarrow (I_{0..n} \in retro\_int\_process)\ ]
\end{array}
$$

Figure 4.13: Predicate calculus definition of retrospective intervallic process.

Formal definition of retrospective registral process: (VP)

(1) All intervals in such a process need to be a perfect fifth or larger (except the first which

may be a diminished fifth).

(2) The intervals are in a sequence of increasing magnitude.

(3) There is less than a major third between any two contiguous intervals.

(4) All intervals have the same registral direction.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (\ (I_n \geq P5) \vee (I_0 = d5)\ ) \wedge & \\
(2) & (\ I_{n-1} < I_n\ ) \wedge & \\
(3) & (\ \mathrm{abs}(I_{n-1} - I_n) < M3\ ) \wedge & \\
(4) & (\ \mathrm{contour}(I_n) = \mathrm{contour}(\ I_0)\ ) & \rightarrow (I_{0..n} \in \mathrm{retro\_reg\_process})\ ]
\end{array}
$$

Figure 4.14: Predicate calculus definition of retrospective registral process.

Formal definition of retrospective reversal: (R)

(1) The initial interval is a minor third, major third or perfect fourth.

(2) The second interval is smaller than the first by at least a minor third.

(3) The contour of second interval is different from the first.

$$
\begin{array}{lll}
(1)\ (\forall\ I)\ [& (\ I_0 \in \{m3, M3, P4\}\ ) \wedge & \\
(2) & (\ (I_0 < I_1) \wedge (\mathrm{abs}(I_0 - I_1) \geq m3)\ ) \wedge & \\
(3) & (\ \mathrm{contour}(I_1) \neq \mathrm{contour}(\ I_0)\ ) & \rightarrow (I_{0..1} \in \mathrm{retro\_reversal})\ ]
\end{array}
$$

Figure 4.15: Predicate calculus definition of retrospective reversal.

Formal definition of retrospective intervallic reversal: (IR)

(1) The first interval is a perfect fourth.

(2) The second interval is smaller than the first by at least a major third.

(3) Both intervals have the same registral direction.

(1) $(\forall I)$ [	$(I_0 = P4) \wedge$	
(2)	$((I_0 > I_1) \wedge (abs(I_0 - I_1) \geq M3)) \wedge$	
(3)	$(contour(I_1) = contour(I_0))$	$\rightarrow (I_{0..1} \in retro\_int\_reversal)$]

Figure 4.16: Predicate calculus definition of retrospective intervallic reversal.

Formal definition of retrospective registral reversal: (VR)

(1) The first interval is either an augmented fourth, an octave, or an interval of a perfect fourth or less.

(2) The second interval us a perfect fifth or greater.

(3) The first interval is smaller than the second by at least a minor third.

(4) The contour of second interval is different from the first.

(1) $(\forall I)$ [	$((I_0 = aug4) \vee (I_0 = P8) \vee (I_0 \leq P4)) \wedge$	
(2)	$(I_1 \geq P5) \wedge$	
(3)	$((I_0 < I_1) \wedge (abs(I_0 - I_1) \geq m3)) \wedge$	
(4)	$(contour(I_1) \neq contour(I_0))$	$\rightarrow (I_{0..1} \in retro\_reg\_reversal)$]

Figure 4.17: Predicate calculus definition of retrospective registral reversal.

4.5. Conclusions

We have critically described, clarified, extended and formalised Narmour's theory. A number of gaps, ambiguities and one inconsistency has been found. These findings can be summarised as follows:

- a predicate calculus notation has been introduced where Narmour's descriptions are unclear,
- a possible inconsistency has been identified in Narmour's use of hierarchical promotion of notes,
- a number of measures of closure have been identified (i.e. metric and harmonic) for which Narmour offers no definition, and in this chapter, and the next, offer explicit, unambiguous measures for such closure,

- another form of closure (due to style) is also not defined by Narmour, but not formalised in our modified version of Narmour's theory since the research to fully model style recognition and learning is a major research project beyond the scope of this thesis.

A structure for which we have not included a definition is retrospective duplication [(D)]. There appears to be a problem with the existence of retrospective duplication as a low-level structure. Surprisingly retrospective duplication, as Narmour defines it, can only occur through the effect of top down style — in which case the phenomenon should not be included in the definition of low level structures. In the second instance Narmour suggests that the listener would not be expecting the notes at an emerging hierarchical level to form a duplication — this implies that the listener is reflecting upon (and generating implications or expectations) the outcome of hierarchical promotion. Again, we would suggest this kind of higher level activity has no place to be modeled as a low level structure. In both cases the concept of high level knowledge effecting the low level structuring of notes is inconsistent with Narmour's claims of an independently functioning, hardwired perceptual system, about which the listener cannot introspect.

Although we have identified a number of concepts requiring formalisation, and have suggested ways to fill in such gaps as harmonic and metric closure, the core rules of inference and parametric scales have needed no additional formalisation. Narmour's theory, as expressed in his writings and hand-produced analyses, has been found to be sufficiently well described that his unchanged inference rules and melodic primitives, along with our formalisations for different kinds of closure, structures and methods for parsing and hierarchical promotion have produced a formalised version that closely matches the theory as he has been applying it to the analyses presented in his numerous examples. Chapter 6 presents a description of how we tested many aspects of Narmour's theory through the comparison and analysis of melodies hand-parsed by Narmour, and those parsed by our implementation of our formalised theory.

Having identified the gaps and ambiguities listed above, we now go on to the next chapter to describe an implementation of a computational model of the formalised version of Narmour's theory presented in this chapter.

Chapter 5:
The parser

> *"..., theories of the mind should be expressed in a form that can be modeled in a computer program. A theory may fail to satisfy this criterion for several reasons: it may be radically incomplete; it may rely on a process that is not computable; it may be inconsistent, incoherent, or, like a mystical doctrine, take so much for granted that it is understood only by its adherents."* (Johnson-Laird 1988, p. 52)

This chapter describes what we believe to be the first computational implementation of Narmour's theory[1]. The computational model, called M-PARSER, takes as input a melody, and produces a hierarchical analysis, based on the clarified, extended and semi-formal version of Narmour's *Implication-Realisation Model* described in the previous chapter.

The development of a parser, to analyse melodies in terms of Narmour's theory, provided two main benefits for the formalisation of the theory:

- the parser provides evidence that the theory has been formalised in an unambiguous representation,
- it has allowed melodies to be input to the parser, and the resulting analyses compared with hand-produced analyses published by Narmour (see Chapter 6),

1 Some connectionist work was begun on the *Implication-Realisation Model*, but the research changed direction (Griffith, 1994), and no model was actually developed.

- when analyses match: it is evidence that the clarified theory has been encoded true to Narmour's informal descriptions, and evidence is provided for the consistency of the theory,
- when analyses do not match: it has allowed the tracking down of gaps, ambiguities and inconsistencies in Narmour's theory, and appropriate extensions or corrections of such representations in the clarified theory,

- a third benefit of having a computational model of the theory is that it can then be extended into a tool for melody analysis and composition, as explored in Chapter 7 of this thesis.

The following chapter (Chapter 6) discusses in detail the comparison of M-PARSER's output with Narmour's published analyses, and discusses the testing of M-PASER these comparisons demonstrate (within the framework of assumptions and simplifications made during the formalisation).

5.1. The structure of this chapter

M-PARSER is presented in five sections as follow (section numbers given in parentheses):

(§5.2) description of the data structures for representing a melody,

(§5.3) description of the data structures for representing an analysis,

(§5.4) a summary of how aspects of the *Implication-Realisation Model* have been declaratively encoded,

(§5.5) an outline of the controlling routines of the parser (which apply the *Implication-Realisation Model* to a given melody),

(§5.6) a step-by-step description of how the parser generates an analysis,

This chapter ends with a summary of those aspects of the clarified version of the *Implication-Realisation Model* that have and have not been computationally modelled, general conclusions and a critical evaluation of M-PARSER.

The relationships between the four programming aspects of the parser (the first four of the above points) are illustrated in Figure 5.1.

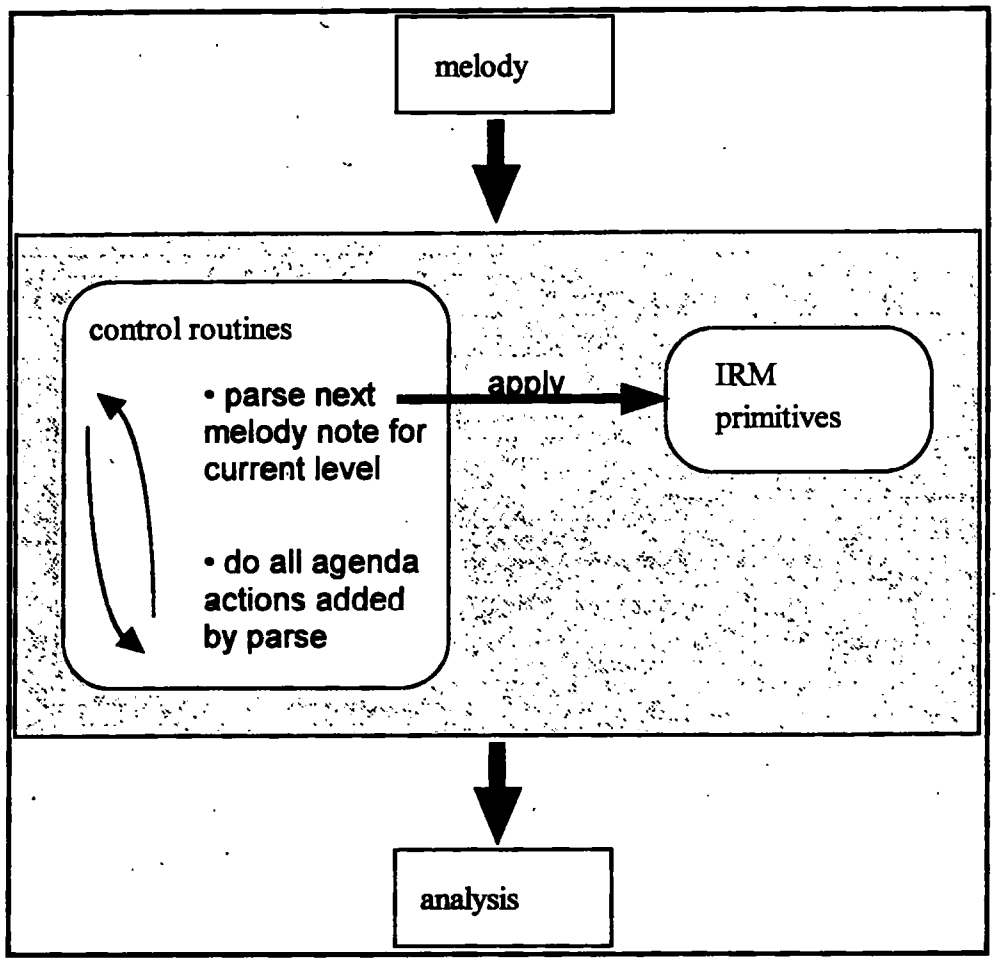

Figure 5.1: Relationship of the four elements of the M-PARSER.

The parser is in the form of two modules (programmed in Prolog): first, a declarative representation of the structural aspects of the *Implication-Realisation Model* (symbolised by the box "IRM primitives" in Figure 5.1); and second, a set of higher level parsing routines. These parsing routines control the application of Narmour's theory to the notes of an input melody in chronological sequence, and the promotion of notes to higher hierarchical levels.

5.2. Representation of melodies

In MOTIVE simple representations have been used for computationally describing melodies; for example an integer for each note's unique identifier, and note pitches represented by an ordered tuple of pitch class, octave number and a list of accidentals. A simple representation of

time (MOTIVE Time Units[2]) allows melodies of varying time signatures and metres to be represented in a standard way. The representations used in M-PARSER have been developed for simplicity[3] — for more sophisticated systems general purpose music representations could be used (such as those proposed by Smaill et al. 1994, and Balaban 1992). For precise details of how melodies are represented in MOTIVE see Appendix D.

Figure 5.2 below shows a simple melody, and Figure 5.3 illustrates M-PARSER's representation of that melody.

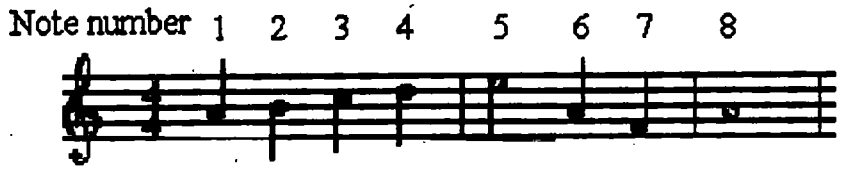

Figure 5.2: A simple melody.

As can be seen in Figure 5.3, at the highest level a melody is represented by a list of unique note numbers (statement (1) in the listing)[4]. Each note (statements (2) .. (9)) has details of pitch (both as MIDI number, and as pitch class, octave and accidental parameters), onset time, duration, and a final argument for any extra information about the note — such extra information might, for example, be used for modelling any extra stress on a note during performances that an analyst considers an additional contribution to closure on a note, or as a way of noting the top-down influence of style on a note (see sub-section 5.4.3). Statements (10) and (11) are lists of the harmonic framework of the melody — i.e. the onset times and durations of keys and modes, and chords that the analyst considers the melody to be heard in terms of. The final statement (12) is a list representing any changes of metre during the melody, each

---

2 MOTIVE Time Units (MTUs) are a simple representational system for time, based on numbers allowing factorisation by the primes 2, 3 and 5 (the basis of almost all musical time signatures) — MTUs are more fully described in Appendix C.

3 This chapter describes the first implementation of M-PARSER, which in some respects is a prototype; so simplicity was the main criterion for the choice of representations (given that the representations chosen also possess sufficient expressivity for the requirements of M-PARSER).

4 The statement numbers (1) .. (12) have been added in Figure 4.3 to aid of exposition.

element in the list storing the onset time and duration of each time signature (both in traditional form and as a metric vector[5]).

The representation of a melody as a list of notes and associated onset times, and of scales and metres in a similar fashion, has been based upon the work of Levitt (1985). In addition, the representation of a chord or scale as a root and associated list of semitone intervals from the root has been adopted (thus a C major chord is represented as `[c, [0, 3, 7] ]`, this is also based on Levitt's work).

```
%melody_list( Note_list )
(1) melody_list([1,2,3,4,5,6,7,8,end]).
%note( note_num, Pitch, Time_point, Duration, Special )
%Pitch = pitch(Pitch_class,Octave, Accidental_list,MIDI_pitch)
%% bar1 %%
(2) note( 1, pitch(a,4,[nat],69), 0, 132300, Special ).
(3) note( 2, pitch(b,4,[nat],71), 132300, 132300, Special ).
(4) note( 3, pitch(c,4,[nat],72), 264600, 132300, Special ).
(5) note( 4, pitch(d,4,[nat],74), 396900, 132300, Special ).
%% bar2 %%
(6) note( 5, pitch(e,4,[nat],76), 529200, 264600, Special ).
(7) note( 6, pitch(a,4,[nat],69), 793800, 132300, Special ).
(8) note( 7, pitch(e,4,[nat],65), 926100, 132300, Special ).
%% bar3 %%
(9) note( 8, pitch(a,4,[nat],69), 1058400, 529200, Special ).
%% for a 4/4 metre
%semibreve (e.g. note 8) = 529200, minim (eg. note 5) = 264600
%crotchet (e.g. note 1) = 132300, quaver (no example) = 66150
% scale_list(List_of [Onset_time, Duration, Root,
   Scale_type]).
(10) scale_list([[   0,1587600,a,aeolian   ]]).
% chord_list(List_of [Onset_time, Duration, Root,
   Chord_type]).
(11) chord_list([[   0,1587600,a,minor   ]]).
% metre_list( List_of [Onset_time, Duration, Metric_vector] )
%% Metric_vector = [Num_beats, Base_note,
   Hierarchical_division_list, Funny_beats]
(12) metre_list([[   0,1587600,[4,4,[1,2,2,2],[]      ]]).
```

Figure 5.3: M-PARSER representation of the melody from Figure 5.2.

5.3. The representation of analyses

An analysis based on the Implication-Realisation Model takes the form of an ordered sequence of levels, each level containing an ordered sequence of structures. M-PARSER represents

---

[5] For details of metric vectors see Appendix D.

analyses as strict hierarchies (each level in the hierarchy is a sequence of structures made up of a subset of the notes in the level immediately below). The structures for a level are contiguous (there is no note for a given level that is not a member of a structure for that level).

In MOTIVE each level represents such attributes as the unique level number, the list of structures for the level, the identifier of the currently active structure for the level, and a list of notes waiting to be analysed for the level.

Each structure of an *Implication-Realisation Model* analysis represents a collection of notes which realise and deny the same implications. When parsing a melody M-PARSER maintains a list of all structures, for each level. Structure attributes represented include the unique identifier (a concatenation of unique level number and the number of the structure for the level), the class of the structure, the implications and realisations of the structure and a list of the identifiers of notes that make up the structure.

5.3.1. An example analysis and its representation in M-PARSER

Figure 5.4 shows part of a melody[6] and its graphical analysis — Figure 5.5 shows the M-PARSER representation for the same analysis[7].

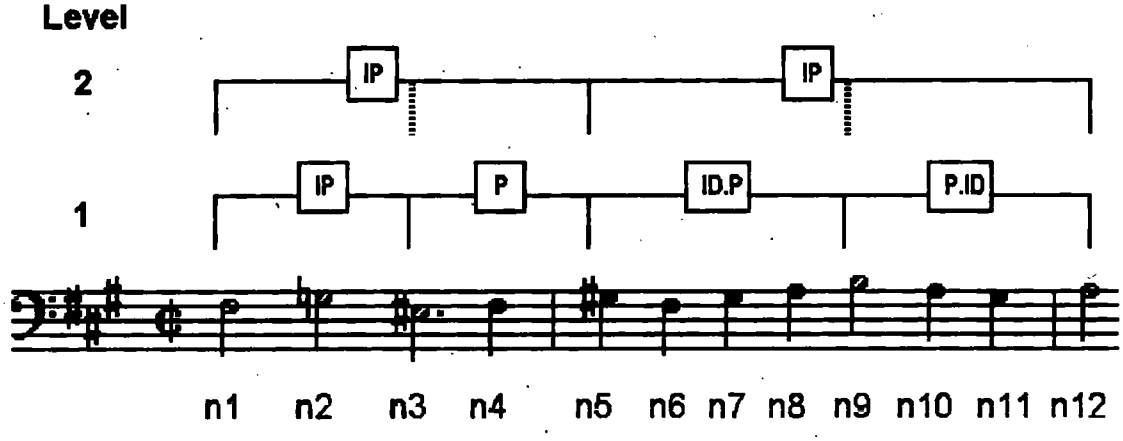

Figure 5.4: A melody and its graphical analysis.

---

[6] Bach, *Mass in B Minor*, Kyrie Eleison II, mm.1-3 — p. 54, Figure 8, Narmour (1989).

[7] In Figures 5.4 and 5.5, to improve legibility of the MOTIVE representation of the analysis, note numbers are given the prefix 'n' (i.e. the first note of the melody is 'n1', the second note 'n2', etc.).

The MOTIVE code presented in Figure 5.5 consists of twelve statements for representing the analysis in Figure 5.4[8]. The first two statements (statements (1) and (2)) show how each of the two levels of the analysis is represented by a list of structure numbers and a list of the notes still to be parsed for each level. Each structure (or part of a chain) has stored about it details of structure class, implications and a list of the notes it encompassed (statements (3) .. (12)).

```
%% Levels %%
%% level(Level_num, Promoted_list, Special,
   List_complete_structs,
      Curr_struct, Unparsed_notes_for_level)
(1) level(1,[s(1),s(2),ch(3,[a,b]),ch(4,[a,b])],[5,new],
      [n12,n13,n14,...]).
(2) level(2,[s(1),s(2)],[3,new],[n12]).

%% Structs %%
%% struct(Level,Struct,Struct_class,Implications,Notes)
%% Implications = (cont/rev,Int_imps,Contour_imps)
(3) struct(1,1,IP, imps(continuation, similarity,
                        [ascent,descent]),[n1,n2,n3]).
(4) struct(1,2,P,imps(continuation, similarity,
                        [ascent]),[n3,n4,n5]).
(5) struct(1,[3,a],ID,imps(continuation,
      similarity, [ascent,descent]), [n5,n6,n7]).
(6) struct(1,[3,b],P,imps(continuation, similarity,
                        [ascent]),[n6,n7,n8,n9]).
(7) struct(1,[4,a],P,imps(continuation, similarity,
                        [descent]),[n9,n10,n11]).
(8) struct(1,[4,b],ID,imps(continuation,
            similarity, [ascent,descent]),[n10,n11,n12]).
(9) struct(1,5,_,_,[]).
(10) struct(2,1,IP,imps(continuation, similarity,
                        [ascent,descent]),[n1,n3,n5]).
(11) struct(2,2,IP,imps(continuation, similarity,
                        [ascent,descent]),[n5,n9,12]).
(12) struct(2,3,_,_,[]).
```

Figure 5.5: M-PARSER representation for the analysis in Figure 5.4.

A detailed description of MOTIVE's representation of analyses can be found in Appendix E. Single structures are represented in the level/4 clause list as elements in the form "s(N)", where "N" is the structure number (e.g. "s(1)", the first structure, IP, at level 1); chained structures are in the form "ch(N,L)" where "N" is the number of the overall chained structure, and "L" is a list of index letters for each of the sub-structures making up the chain — e.g. "ch(3,[a,b]",

[8] The statement numbers (1) .. (12) have been added to the figure for ease of exposition.

representing the third structure at level 1, of "ID" and "P", with individual facts of "struct(1,[3,a],...)" and "struct(1,[3,b],...)"[9].

5.4. The Implication-Realisation Model primitives of M-PARSER

In addition to the representation of melodies (the input to the parser) and analyses (the output of the parser), the basic concepts of the *Implication-Realisation Model* are represented in the parser — for example the scales for pitch, contour and duration, and the measurements of closure relating to the positioning of a note or interval's parameters on such scales.

This chapter provides a brief overview of how the theory (as described in the previous chapter) has been computationally encoded. More technical detail can be found in Appendix F.

5.4.1. The parametric scales

The parametric scales described in the previous chapter are modelled as lists in M-PARSER. Two values for a given melodic parameter can then be compared in terms of their position in the appropriate scale list, and the strength and form of closure or non-closure can be calculated. The representation of the scales also include definition of boundaries (such as between the major third and perfect fourth in the intervallic scale), and of special elements in the list (such as the octave in the intervallic scale).

5.4.2. Implication leading from the rules of inference

The size of the generating interval is initially measured in terms of the boundaries of large and small intervals on the parametric scale for interval. Once the class of generating interval has been determined, the implications leading from the interval class are recorded for the active structure. At this point the parser may determine that the two notes making up the interval are a dyad, and the structure is tested for strength of closure.

If a second interval is parsed, the set of prospective structures associated with the generating interval's implications is interrogated. If a match is found, the structure's implications,

---

[9] The underscores for the structure class implications in (for example statement (9)) represent parts of current structures that are not known, or parts of structures not yet included at a point in time during parsing.

realisations and denials are recorded for the active structure. If no prospective structure match is found then a retrospective structure must have occurred, the retrospective implications are determined, and the set of associated retrospective structures is interrogated.

5.4.3. Top-down stylistic learning

Narmour refers to the top-down influence of stylistic learning, and in many cases closure of structures is attributed (in part) to stylistic causes — either from style within the melody (*intra-opus* style), or style from a collection of melodies or school of music (*extra-opus* style).

No automatic, top-down stylistic routines have been implemented, however, the melody representation has been designed to allow analysts to include information about special causes of closure on any note of a melody, and it is in this way that the influence of style can be incorporated into the parsing process of MOTIVE.

5.4.4. Closure

The degree of closure on a note is represented by an integer value; in the present implementation of M-PARSER this value is determined by simple addition of the integer values for each form of closure multiplied by its associated weighting factor. As described above, closure from top-down stylistic influence is only modelled if explicitly assigned to a note by an analyst (once again, the degree of closure due to top-down style is represented as an integer). The other forms of closure modelled are as follows.

Melodic closure

Melodic closure occurs (to some degree) whenever there is a change of interval size or of contour. The melodic parameters of interval size and contour are measured using Narmour's parametric scales. An integer representing the number of steps along the scales, and direction (i.e. positive if movement left on a scale, zero or negative otherwise) between the values for a sequence of two intervals determines closure due to melodic parameters.

Durational closure

As with melodic closure, Narmour's scale for duration is used to determine the degree (number of steps along scale) and form of closure (i.e. positive if movement left, when an interval is from a short note to a long note). As discussed in the previous chapter, when a short note is

followed by one of duration 1.5 times or more, durational closure of transformational strength occurs.

Metric closure

Metric closure is measured in terms of how the onset time of a note can be factorised in terms of the time signature for the part of the melody in which the note occurs. The time signature is represented in terms of metric vectors (see Appendix D), which allow the factorisation of the onset time into the total duration of the bar (both onset time and bar duration are represented as MOTIVE Time Unit's, see Appendix C). The result is that for any two notes, two rational numbers can be calculated (using only denominators determined from the time signature), the difference in denominators determines the strength of closure. The form of closure is positive if the second note of the two has a denominator to the left of that of the first note, for the ordered list of demoninators defining the current metric vector.

Harmonic closure

In the current implementation of M-PARSER harmonic closure is modelled in a very simple fashion. Each note of the melody has a value assigned to it, determined by which of Lerdahl's (1988) pitch spaces the note belongs, in terms of the active scale (mode or key) at the point in the melody where each note occurs. A scale is represented, with level one ("tonic space", the most consonant) at the leftmost end, and level five ("chromatic space", least consonant) at the rightmost end. Once again, movement left on this scale determines positive closure, and the number of steps determines magnitude. See the further research section of Chapter 8 for discussion of how the measurement of harmonic closure could be made more sophisticated by more detailed modelling of harmonic relationships based on Lerdahl's pitch spaces.

Closure due to stopping

One way a melody "stops" is when significant rests are met. In the current implementation of M-PARSER a significant rest is one which is more than half the duration of the preceding note. If significant, the number of steps along the duration scale from the duration of the note to the duration of the rest determines the strength of closure due to stopping. Another form of stopping is when the end of the melody has been reached; in the current implementation of M-PARSER this results in the preceding note being given a retrospective value of closure due to stopping magnitude of 3 (initially the arbitrary value of 1 was used, but it was found the final note of a melody is often promoted due to more significant closure due to stopping —

a value of 3 is used in the current implementation of M-PARSER, which informal testing has shown to be a value leading to greater matching of final note promotion in analyses by M-PARSER and Narmour's published analyses). Stopping is also the term Narmour uses for when a prospective structure encounters a note that does not match its implications. The structure must terminate at this point (although any promotion and merger will be according to the degree of forms of closure).

The weighting of closure strengths

The weighting factors were initially arbitrarily set to an equal value of 1 each. However, initial testing showed that closure due to metre and duration was not influencing the parsing process in M-PARSER as significantly as in Narmour's published analyses, so the weighting factors for these two forms of closure have been set to values of 2 and 3, for metric and durational closure respectively. Methods of "fine-tuning" M-PARSER's closure and weighting factors are described in the further research section of Chapter 8.

5.5. The control routines of M-PARSER

Program code has been developed to automate the procedural aspects of parsing that human music analysts perform (such as choice of next note to consider, action to take on next note — does it start a new structure, has an ongoing structure been terminated, etc.). These routines are referred to in this thesis as the *control routines* for M-PARSER, and are described below. The control routines use a simplified form of forward chaining with an agenda. The next four subsections describe the control routines as follows (subsection numbers in parentheses):

§5.5.1. **the parsing algorithm** — the overall parsing algorithm for M-PARSER is described via pseudo-code,

§5.5.2. **parser cases** — forward chaining ("`parse/5`") routines apply the primitives of the Implication-Realisation Model to the current note for the currently active level of the analysis, and then the appropriate clause fires and will add a sequence of actions to the agenda,

§5.5.3. **agenda actions** — for each action that can be added to the agenda there is a "`meta_action/6`" clause, which performs the specified action, and

§5.5.4. **parsing sequence** — a brief description of the sequence of events occurring when a melody is parsed by the parser cases and agenda action routines

First, the different actions that the parser takes are briefly described; second, the sequence of steps in the parsing process are outlined, with reference to when (and which) parser actions are performed.

5.5.1. The parsing algorithm

The parser works in a modified forward chaining way, using an agenda. The general algorithm is summarised in pseudocode in Figure 5.6. At the start of the program (START in Figure 5.6) the melody is loaded (step 1), and an empty analysis is created (step 2), with all notes of the melody assigned to the lowest level in the analysis hierarchy. As notes are parsed for the lowest level of the analysis, some will be promoted to the next higher level, thus starting the recursive application of analysis at other levels. Next an empty agenda is created (step 3), and control passed onto the main forward chaining parsing sequence. The main processing of the parser is done by the routines called "LOOP" in the Figure 5.6 — these routines implement a very simple form of forward chaining, whereby each time the agenda is empty (step 5) the parser routines examine the structure that is ongoing for whichever level of the analysis is currently active. One of the parser routines will always match, and add one or more actions to the agenda. While there are (non-termination of parser) actions in the agenda (step 4) they are performed in sequence — the parse routines are not called again until the agenda is empty. The program finishes when a parse routine creates an agenda containing the single element "end of parse" (the exit condition for the loop).

```
START
        1   load melody
        2   create an empty analysis, with all notes of melody assigned to lowest level
        3   create an empty agenda

LOOP WHILE agenda not equal to [ end_of_parse ] DO
        BEGIN CASE of agenda state
            4   IF agenda has at least one AGENDA_ACTION action in it THEN
                        do the first AGENDA_ACTION in the agenda
            5   IF agenda is empty THEN
                        PARSE melody to generate new agenda items
        END CASE
END LOOP
```

Figure 5.6: Pseudocode describing parser algorithm.

The parts of the parser that are not described in detail in Figure 5.6 are the parse routines (PARSE), and the different actions that are added to the agenda (AGENDA_ACTIONS). The two sets of routines are summarised below.

5.5.2. The parser cases

Figure 5.7 below lists the fifteen different cases the parse routines identify for the state of a melody being analysed. The figure is in two columns, the left column describing possible state descriptions, and the right column listing the actions to be added to the agenda if all the conditions for a parser rule are satisfied — only one parse rule will fire each time the parse routines are called, and the rules are tested in the order given below (there is no conflict resolution for possible multiple rule firings).

State of analysis		*Actions for agenda*
1	top_level + end_of_melody	[end_of_parse]
2	top_level + not end_of_melody	[descend]
3	ongoing_structure + not top_level + end_of_melody	[close curr struct(nochain), promote(Last), promote(end_of_melody), ascend]
4	new_structure + not top_level + end_of_melody	[promote(end_of melody), ascend]
5	note notes for current level	[descend]
6	new struct	[make ongoing]
7	empty ongoing_struct	[append(Next)]
8 / 12	SEPARATE closure	[close_curr_struct(nochain), promote(Last), ascend]
9 / 13	SHARE closure	[close_curr_struct(nochain), copy_notes(Last), promote(Last), ascend]
10 / 14	MERGE closure	[merge_structure, close_curr_struct(chain), copy_notes(Second_last, Last)]
11	next note makes a retro structure	[append_next, make_retro(Struct_class,Imps)]
15	NEGLIGIBLE closure	[append_next]

Figure 5.7: Pseudocode description of cases for parse routines.

The agenda actions are described in the next section. The conditions for the parse rules have been described using the following abbreviations:

- **top_level** this state is when the current level to be analysed is the top level (one of the preference variables for M-PARSER is the number of levels to which a melody should be analysed)

- **end_of_melody** this states that there are not more notes in the melody to be parsed for the current level (i.e. the sentinel note "end_of_melody" is the next note

in the list for the current level),

- **ongoing_structure** the current structure is ongoing,

- **new_structure** the current structure for the current level is an empty, new structure,

- **closure** the form of closure on the current structure for the current level — first any form of prospective closure is tested (rules 8, 9, and 10), on the most recent note to be added to the melody, if no prospective closure is present a test for any form of retrospective closure is made (i.e. closure due to the next note to be processed for the current level — steps 12, 13, and 14). M-PARSER classifies four forms of closure strength:

  (1) SEPARATE — closure causing the current structure to terminate and promotion of its terminal note, and to be completely separate from the next structure for the level,

  (2) SHARE — closure causing the current structure to terminate and promotion of its terminal note, and to share its terminal note with the next structure for the level,

  (3) MERGE — closure causing the current structure to terminate, but with no promotion, and to share its last TWO notes (i.e. its last interval) with the next structure for the level[10],

  (4) NEGLIGIBLE closure — where the current structure has the next note for the level appended to it, and continues as an ongoing structure (where either there is no closure, or any that is present is not sufficient to effect the currently active structure), and

- **next note makes retro structure** when the next note for the level is appended to those for the current structure, the resulting structure is a retrospective structure.

---

10 This is the parser rule for identifying combinations and chains, as described in the previous chapter.

5.5.3. The agenda actions

M-PARSER's agenda consists of an ordered list of the following actions:

- **`end_of_parse`** the parse is complete,
- **`ascend`** ascend to next highest level in hierarchy, and work on the active structure at the new level,
- **`descend`** descend to next lowest level in hierarchy, and work on the active structure at the new level,
- **`close_curr_struct(chain/nochain)`** close the active structure, and create a new, empty structure for the current level (if the "chain" option is given, the new structure will be the next for the chain, i.e. 'b', 'c', etc.),
- **`promote(Note)`** promote the named note to the next highest level in the hierarchy (the named note is one of: Last, Next, end_of_melody, being the last note for the current structure, the next note to be parsed for the current level, and the sentinel note marking the end of the melody respectively),
- **`make_ongoing`** make the current structure ongoing (this action is executed when the parser begins to work at a level with an empty active structure),
- **`copy_note(Note_list)`** add a copy of the given notes of the current structure to the front of the list of notes for the current level (used when a structure is closed, but will share these notes with the next structure for the level),
- **`merge_structure`** make the current structure one of a chain of merged structures (i.e. if the current structure is not part of a chain, create a new chain and change the details of the current structure to be the first of the chain), and
- **`append_next`** append the next note for the current level to the current structure for that level.

The sequence of steps taken by the control routines of the parser are described in the next sub-section, and refer to the above agenda actions. The actions are notated as follows: **`[<action_name>]`**.

5.5.4. The parsing sequence

Although the parsing sequence is encoded in the declarative form described in the two sections above, it is useful to describe the sequence of actions that the parsing process follows as a *sequence*. M-PARSER begins working at level 1 (the level immediately above the musical surface). This level is given a list of all the notes in the melody — the remaining (higher) levels will only parse notes that are promoted initially from level 1. For each level an active structure is represented. At any time a level is in one of three states:

- a "new" structure is the current one for the level (empty of notes, and inactive) (either the first structure for the level, or because of the closure of the previous structure),
- there are no more notes to be parsed for the current level (either the end of the melody has been reached, or the next level below needs to be parsed more — to provide new notes for the current level via note promotion), or
- the next note should be analysed with respect to the active structure for the current level.

The first and second cases are relatively straightforward — an empty, ongoing structure can be created and the next note for the level assigned to it **[append_next]**. If there are no more notes for the current level, either the end of the melody has been reached **[end_of_parse]**, or there the next lower level in the analysis should be parsed some more to result in more notes being promoted, i.e. the M-PARSER should make the next lower level in the analysis the active level **[descend]** .

In the third case above, when there is an existing, ongoing structure, M-PARSER looks at the last two notes of that structure with respect to the next note for the level, and processes them in a number of steps. Since prospective closure is tested for before retrospective closure, the steps below will be performed initially on the last two[11] notes of the current structure, and if no prospective closure is found, the steps will be performed a second time on the last note of the melody and the next note for the level.

---

[11] If the current structure has only a single note in it, then only tests for retrospective closure are made.

The steps of the parsing algorithm are as follows:

1 Check note attributes (pitch, onset time, duration).

2 Calculate attributes of interval between new note and previous note (interval size in semitones, contour).

3 Check whether in the current context the interval is considered large or small according to the implication-realisation model (small intervals initiate continuation expectations, while large ones initiate reversal)

4 Check if stylistic knowledge matches with the current events and if so, generate expectations according to a style structure (not yet implemented).

5 If non-negligible closure has occurred, close the current structure **[close_curr_struct]** (promoting a note **[promote(Note)]**, appending the next level note **[append_next]**, making it a merge structure **[merge_structure]**, and copying the last one **[copy_last_note]** or two **[copy_last_two_notes]** notes of the current structure to the current level's notes if appropriate),

6 Run an intra-opus learning program (not yet implemented).

7 If negligible closure has occurred, simply append the next note **[append_next]** for the level to the current structure,

8 If promotion of a note has occurred, then ascend **[ascend]** to the next higher hierarchical level and go through steps 1-8 for the notes and structures at the next hierarchical level.

As M-PARSER moves through the melody it builds up structures at a number of hierarchical levels, and exhibits the re-structuring of past notes when it encounters situations of retrospective structures. Although a more efficient parser could be written, working on a complete melody in one pass, it would have the limitations of not being able to show the intermediate structures before retrospective structures are identified, and could only work on complete melodies. The note-by-note parser described in steps 1-7 above (i.e. M-PARSER) has neither of these limitations.

5.6. Step-by-step example of parsing

5.6.1. Overview of the step-by-step parse

In this section a step-by-step breakdown of the parsing actions involved for an analysis is presented. The melody and resultant analysis is that illustrated previously in Figures 5.4 and 5.5 (Bach, *Mass in B Minor*, Kyrie Eleison II, measures 1-3, as analysed by Narmour 1989, p. 54).

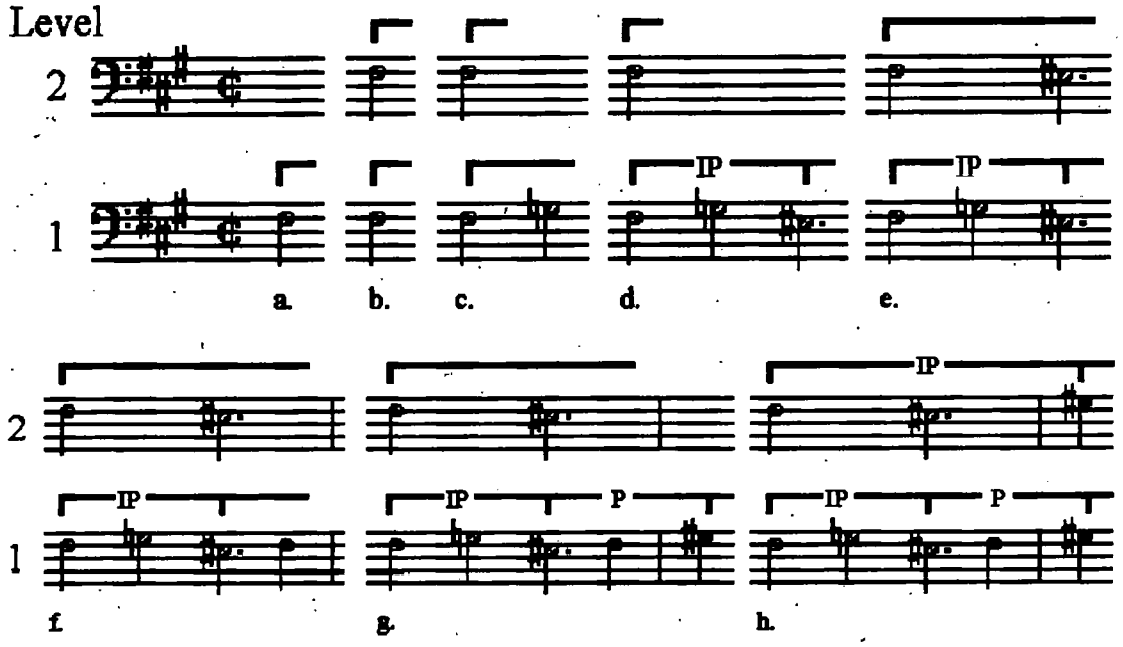

Figure 5.8: Breakdown of parsing stages in CMN.

The description of the step-by-step workings of the parser is presented in two stages. The first stage consists of steps ***a.*** - ***h.*** of Figure 5.8 — illustrating in common music notation how the first 5 notes of the melody is parsed and some notes promoted to the next higher level in the hierarchy. The parsing of the first five notes contains examples of how structures are started, terminated, and notes promoted to higher levels to be parsed recursively by the same routines. The second stage describes the analysis of notes n5 - n9. These notes when parsed illustrate the formation of a combined structure due to the merging of an intervallic duplication **[ID]** and a process **[P]** structure. This stage is illustrated by the steps ***a.*** - ***e.*** in Figure 5.9 (step ***f.*** is explained at the end of the section describing stage 2). Each of the parser steps for the two stages are described below.

5.6.2. Stage 1: Description of the analysis of first 5 notes of melody

The representation of an analysis is initialised to a set of empty structures, and a set of levels each with an empty list of completed structures, with the current structure set to the first for the level with the status of a "new" structure. The current level is initialised to level 1 (the first level of analysis above the musical surface), and for this level all notes of the melody are placed in the list of notes for parsing for the level (all other levels start off with empty lists of notes for parsing).

When parsing begins the action agenda is empty so the parse routines are called. These routines spot that the current structure for the current level has a "new" status, so the single action "**make_ongoing**" is placed in the agenda and subsequently executed.

(Step a.) When the parse routines are next called, they identify that the current structure for the current level has no notes associated with it, and that there are unparsed notes for the current level. This leads to the placing of the action "**append_next**" being placed in the agenda and executed[12]. The parse routines also identify that the note just added to the current structure is the first for the structure, and that the structure is not a continuation of a sequence of structures sharing any notes or intervals. This leads to the actions "**promote(** ***n1*** **)**" and "**ascend**" being added to the agenda. When these actions are executed the analysis status is that at level 1 there is a single structure with a single note, and notes ***n2*** - ***n12*** remain to be parsed. At level 2 there is a single notes to be parsed (the newly promoted n1), and that level 2 is now the current level (this is the result of the "ascend" action being executed when the previous current level was level 1). This leads to the state of the analysis as illustrated by step ***a.*** of Figure 5.8 (in terms of structures of the analysis only the note in the structure at level 1 has been completed parsed). The structure at level 1 is "ongoing", but as yet has not implications since it only contains a single note. The structure at level 2 is currently empty, and has status "new".

(step b.) The parse routines will follow the same sequence for the first structure of level 2 (**make_ongoing, append_next**), and if the parser were set up to parse to higher levels would

---

[12] Note that when the agenda action "append_next" is executed the first note in the list of notes to be parsed for the current level is remove from that list, and appended to the end of the notes for the current structure for the current level.

also add the actions "**promote(*n1*)**" and "**ascend**". For this example we will assume that the maximum number of levels for parsing has been set at 2, therefore once note **n1** has been appended to the first structure of level 2 there are no more notes left at that level for parsing. This will be spotted by the parse routines when they are next called, and the "**descend**" action will be placed in the agenda and then executed. The result being that the current level returns to being level 1.

(step c.) The parse routines work on the ongoing structure at level 1, examining the next available note for parsing (note **n2**) for retrospective closure (i.e. seeing if note **n2** retrospectively causes the first structure to be a monad). This is not the case (there is no significant metric, harmonic or durational closure due to **n2**). Since this first structure has no implications yet, any note can be appended to the structure in the absence of closure, so the agenda action "**append_next**" is added to the agenda and then executed (once again after the appending of a note to the structure a check is made for prospective closure, which is absent on note **n2**). This results in the analysis status as described by step *c.* in Figure 5.8 — the current level is level 1, there is one, ongoing structure containing notes **[n1, n2]**, the remaining notes of the melody (**[n3 .. n12]**) are waiting to be parsed for level 1; at level 2 there is a single ongoing structure, containing note **n1**, and there are no other notes at level 2 waiting to be parsed.

(step d.) The two notes in the ongoing structure at level 1 generate implications of continuation (due to the small interval between notes **n1** and **n2**). The parse routines look at the next note to be parsed for the level (note **n3**) and conclude that although there is a change of contour in the intervals from **n1 - n2** and **n2 - n3**, the interval between **n2** and **n3** is small, therefore **n3** can be appended to the ongoing structure (**append_next**), and the structure identified as an intervallic process **[IP]**. The parse routines then test for prospective closure. On note **n3** there is both durational (minim to dotted-minim) and metric closure (beat strength of 2 to 1, or onsets of ¼ bar to ½ bar). The closure is strong enough to terminate the structure and cause a promotion of the terminal note to the next level in the hierarchy, However, the closure is not sufficient for "separate" closure; i.e. the result is that the **[IP]** is closed, but the final note, **n3**, will form the first note of the next structure. The parse routines when identifying this situation place the following actions in the agenda:

[close_curr_struct(nochain), copy_notes([n3]), promote(n3), ascend][13]

The result of these actions is illustrated by step *d.* of Figure 5.8, where at level 1 there is one complete **[IP]** structure, and an empty, new structure (about to contain note **n3**), the current level is level 2, and note **n3** has just been promoted into the list of notes for parsing at level 2 (although currently the only completed note parsed at level 2 is note **n1**).

(step e.) The parse routines, working at level 2, simple add the agenda action "**append_next**" to the agenda. This is executed, and note **n3** added to note **n1** in the ongoing structure at level 2. This is illustrated in step *e.* of Figure 5.8. The parse routines then add the agenda action "**descend**", since there are no more notes to parse at level 2 (the result is that level 1 becomes the current level).

(step f.) The current structure at level 1 is a new one (made current when the **[IP]** structure was closed), therefore the parse routines add the agenda action "make_ongoing". After the structure is made ongoing, the parse routines, faced with an empty ongoing structure, simple add the "**append_next**" action to the agenda. This when executed adds the note **n3**. This note, the first of an ongoing structure is not promoted since it shares a note with the previous structure for the level. After a check for prospective closure (of which there is insufficient to make the structure a monad), the parse routines again add the "**append_next**" action to the agenda. After execution the **n4** note has been added to the list of notes for the current structure, and there are implications of continuation for small intervals and an upward contour. There is not sufficient closure to close the structure after the addition of **n4**. This is the situation in step **f.** of Figure 5.8.

(step g.) The parse routines check if given the next note to be parsed for the current level cause the current structure to be retrospectively closed (this is not the case). The next note, **n5**, meets with the implications of continuation, creating a small interval (**n4 - n5**) and an upward contour. Therefore the parse routines add the action "**append_next**" to the agenda, and upon execution note **n5** is added to those for the current structure, which is identified as a process **[P]**. A check for prospective closure is made, and there is some closure from the weak

---

[13] The inclusion of the action "**copy_notes([n3])**" will ensure that the note **n3** is appended to the empty list for the next structure for the level, i.e. this is the implementation of making a single note be shared between two structures when there is insufficient closure for two separate structures.

to strong beat strength (note **n5** has strong beat strength since it is the first note of a bar), although there is no additional closure due to duration or harmony. The result is sufficient closure to terminate the structure with "share" closure; i.e. as with **n3**, the terminal note of this process structure (**n5**) is promoted, but also will be shared with the next structure for the level. The parse routines do this by adding the following actions to the agenda: [**close_curr_struct(nochain), copy_notes([n5]), promote(n5), ascend**]. The result, after these actions are executed is illustrated in step **g.** of Figure 5.8; the process structure is terminated, but its final note is the beginning for the next structure, the current level is level 2, where the current structure has two notes, and in the list of notes to be parsed at level 2 a third note, the newly prompted **n5**, is to be found.

(step h.) A check is made for retrospective closure of the structure at level 2 (which does not occur). The next note for the level, **n5**, meets the implication of a small interval, but denies the implication of a downward contour. The result is another intervallic process [**IP**] structure, which is formed after the execution of the agenda action **"append_next"**. This structure is then tested for prospective closure, which occurs of strength "share" (the closure is due to the beat strength of **n5** being stronger than that of **n3**, but not stronger since there is no additional closure due to duration or harmony). The result is the adding of actions to the agenda for the closing of the structure and copying of the note **n5** to become the first note of the next structure. This final status is illustrated in step **h.** of Figure 5.8, where the current level is level 2, which contains one terminated structure, and has a new structure (which is about to also contain note **n5**), at level 1 there are two terminated structures, and the third structure for the level is about to also contain note **n5**.

5.6.3. Stage 2: Description of the analysis of two merged structures

The steps above have described the general parsing technique of M-PARSER, and illustrates how (via agenda actions) structures are made ongoing, have notes appended to them, are terminated, and how notes are prompted to higher levels for the recursive application of the same rules.

This section compliments the above explanation, by giving a description of the steps involved when M-PARSER analyses structures that form chains or one or more. Figure 5.9 shows the steps of the parsing of notes **n5** - **n9** of the Kyrie melody discussed above, where a chain of two structures is formed (which Narmour would call a combination, since precisely two structures

are involved).

(steps a., b. and c.) In the same manner as described in section 5.6.3, through the parsing routines adding actions to the agenda (**make_ongoing** and **append_next**) notes **n5** - **n7** are added to the current structure. With the addition of **n7** (the G#, third note in step **c.**) the parser identifies the notes of the structure as an intervallic duplication **[ID]**. The sequence of steps is illustrated by steps **a.**, **b.** and **c.** in Figure 5.9. At this point the structure is still ongoing, with implications of intervallic duplication.

(step d.) There is extremely weak closure on note **n7**, the increasing metric differentiation from **n6** to **n7** (onset on an eighth of a bar to onset on a quarter of a bar), this in combination with a simple stopping of the structure (the next note, **n8**, does not meet the implications of an interval of two semitones), results in insufficient closure to properly terminate the structure. The result is that the **[ID]** structure merges with the next structure to be parsed; i.e. the two notes forming the second interval of the **[ID]** structure form the first two notes of the next structure. The parse routines that identify this situation add the following actions to the agenda: **[merge_structure, close_curr_struct(chain), copy_notes([n6, n7])]**. No notes are promoted and therefore there is no need for the parser to ascend to the next higher level to parse new notes at that level. The old structure is renamed from structure 3 to [3, a], to indicate that it is the first in chain (this is the result of the execution of the "**merge_structure**" action). A new structure is created for the level, with the number [3, b] (created by execution of the **close_curr_struct(chain)** action). Once the notes are added to the level, the new structure is made ongoing, and the notes **n6** and **n7** are appended to it. Once the next note for the level, note **n8**, is appended, the structure is identified as a process **[P]**. This is illustrated by the grouping of notes **[n5 .. n8]** as a **[ID.P]** chain in step **d.** of Figure 5.9.

(step e.) There is no stopping, and note **n9** is appended to the process structure in the chain. There is both durational and metric closure on note **n9** (crotchet to minion, an eighth of a bar onset to a half bar onset), also the next note denies the implications of upward contour. The result is that the **[ID.P]** chain is terminated with "share" closure, so that the final note, **n9**, is promoted, and also copied to the notes to parse for the level, so that if forms the first note of the next structure. This state is illustrated by step **f.** of Figure 5.9.

(step f.) The final step of Figure 5.9 (step **f.**) is included to aid understanding of the particular chain structure in the example. From this step of the figure, it is clear how the **[ID.P**

] chain is made up of a merging of an intervallic duplication [**ID**] structure composed of the first three notes (notes **n5**, **n6** and **n7**), and a process [**P**] structure composed of the second two notes of the [**ID**] structure (**n6** and **n7**) and the two notes that follow those (notes **n8** and **n9**).

5.6.4. Summary of step-by-step parser description

It is the aim of the two stage descriptions above to help explain the parsing algorithm and its implementation as M-PARSER. Descriptions of the active levels and status of the construction of structures have been given, in terms of the phenomenon identified by the parse routines, and the resultant actions placed into the agenda and their execution.

5.7. Summary of implementations

This section summarises what aspects of the *Implication-Realisation Model* have been computationally modelled. M-PARSER breaks down into two sets of routines: first, those that declaratively describe the theoretical primitives of Narmour's theory; and second, the control routines which model the sequence of actions performed by a human analyst when applying the *Implication-Realisation Model*. Each of the sets of routines is described in a sub-section below.

5.7.1. Implementation of theoretical primitives

Generally the implementation of primitives in M-PARSER follows the descriptions of the primitives in the previous chapter. The main simplifications and non-implemented aspects of the *Implication-Realisation Model* for M-PARSER can be summarised as follows.

Style recognition

Currently there is no implementation of any kind of intra- or extra- opus style recognition procedures. This is due mainly to the size of the task — the design and implementation of such procedures, along with ones for learning of style (see below), would be a major project for further research (and such research is discussed in Chapter 8).

Style learning

For full implementation of the *Implication-Realisation Model* a set of style learning procedures

would have to be implemented, and run simultaneously with the parser. This would be to allow the learning and application of intra-opus style structures while parsing a melody. We have identified (see earlier this chapter) in the procedural description of the parsing sequence where such procedures would be called in the current parsing framework. Similar procedures could be run "off line" to build up a knowledge base of extra-opus style structures. It would be advantageous if the style structure representation adopted fit well with a constraint formalism, allowing similar routines to be used for constraint inference in MOTIVE (see the further research section of Chapter 8).

Octave intervals

As described in the previous chapter, the current analysis of whole octave leaps has been simplified to consider them as unison intervals. This aspect of the *Implication-Realisation Model* needs to be modified to be more sophisticated, and a corresponding increase in the complexity of M-PARSER implementation effected.

Calculation of combined closure

Currently the method for calculating the closure on a note when it is influenced by a number of forms of closure is a straightforward, weighted calculation — Narmour's most recent book (Narmour, 1992) goes some way to describing the results of combined closure; these aspects of the *Implication-Realisation Model* need to be further clarified and quantified, and then corresponding procedures implemented for them in M-PARSER.

5.7.2. Implementation of control routines

A full set of control routines has been implemented in M-PARSER for the primitives that have been encoded. The only routines that will need adding at a future time are those to call style recognition and learning procedures, and to include stylistic measures in the calculation of type and degree of closure for a given structure.

5.8. Conclusions and critical evaluation of M-PARSER

M-PARSER, a computational model of the bottom-up aspects of Narmour's Implication-Realisation Model has been implemented. It is based on the chronological, note-by-note process of parsing inferred from Narmour's published analyses and descriptions of his theory. In this chapter we have described the structure of the parser, and presented a description of how the

parser processes melodies in a step-by-step fashion, following the inferred procedural technique used by Narmour for his analyses.

Simplifications have been made in the development of the parser, mostly based around the measurement of closure on notes, and the determination of when notes should be promoted in the hierarchical analysis. The simplifications have been largely necessary due to lack of description by Narmour of the how metric, harmonic and style-influenced closure are determined, and lack of formal statements of when closure is sufficient for hierarchical promotion, or weak enough for chaining or combining of structures to occur.

M-PARSER is not based upon a strict grammar used by many natural language programs (for example: Allen 1987, Colmerauer 1978, Heidon 1975, and Sowa 1984). The need for different kinds of analysis of the musical context of different notes (e.g. the harmonic or metric closure, the calculation of combinations of different forms of closure etc.), existence of "retrospective" structures, and the sequence of steps in the parsing process (inferred from Narmour's publications, and implemented as a set of control routines in M-PARSER) has led the development of M-PARSER to be tailor written, rather than adapting existing parsing algorithms such as chart parsers (Baker 1989b, Winograd 1983). An advantage of the parser being tailor-written is that it has been designed with a view to incorporating a number of extensions to both make the parser more sophisticated, and a closer model of Narmour's theory[14] — for example M-PARSER has been written to work in conjunction with a top-down stylistic (or phrase-variation) analysis module, when such a model has been implemented. It is possible that an adaptation of a sophisticated parsing technique could allow for the modelling of the above complexities for the analysis of a melody according to the Implication-Realisation Model. However, the most important reason for the parser being tailor written was that fact that the parser, as well as being for the parsing of melodies, has always been intended to form the basis of a set of constraint-based routines for melody generation, and as such needed to be written in a declarative, modular form, suitable for "plugging in" to a constraint-based generator with the application of additional constraints. In Chapter 7 we described the constraint-based extensions to the parser — it was this need for the melodic primitives, parametric scales, structure definitions, and harmonic and metric closure analysis modules to

[14] Such as the modules for inference of harmonic contexts and metre, discussed in the further research section of Chapter 8.

be expressed in a constraint-satisfaction form that drove the development of M-PARSER into the form described in this chapter.

The development of the parser has been based on the formalisation of Narmour's theory described in the preceding chapter. This has the advantage of M-PARSER being based on an unambiguously expressed theory, but the disadvantage that the proposed formalised version of the *Implication-Realisation Model* involved some decisions in cases of ambiguity, which may mean the formalised theory is a slightly inaccurate description of Narmour's theory. The results of testing the formalised theory described in the next chapter, in addition to future dialogue about M-PARSER with music analysts conversant with the *Implication-Realisation Model*, provide a means of reducing any differences between the theory that M-PARSER embodies, and the theory intended by Narmour. Of course, the identification of ambiguities within the *Implication-Realisation Model* described in the preceding chapter contributes to the theory being made more formal itself. Results from future testing of the psychological validity of aspects of Narmour's theory can also be taken into account when modifying the formalised version of the theory.

Chapter 6:
Testing Narmour's theory

6.1. The aims of the testing

This chapter describes a testing of the formalised version of Narmour's theory described in the previous two chapters. The limitations of M-PARSER, and the implementation of the formalised theory, are also considered. The aim was to test to what extent M-PARSER produces the same analyses as those published by Narmour (applying his analytical theory "by hand"); and to account for any differences. In the previous chapters we have proposed definitions for aspects of the *Implication-Realisation Model* which we identified as incomplete, ambiguous or inconsistent, therefore the testing of our parser against Narmour's application of the unformalised version of his theory can only form part of the testing of a formalised version of the theory — our parser should not to be able to reproduce the subset of Narmour's analyses based on those parts of his theory we have proposed changes to. As suggested by Baker (1995), a full testing of our formalised version of the theory would need to have a cognitive basis, i.e. an evaluation of our theory (and parser) in terms of how well its analyses model those built up by listeners (either novices or expert musicians and musicologists) from a cognitive psychology of music perspective. Some issues raised by these questions are discussed in the further work section of the final chapter of this thesis. The test results presented in this chapter aim to demonstrate how the fundamentals of Narmour's theory (which have not been changed in our formalisation) have been formalised and computationally implemented in M-PARSER.

Discrepancies between the analyses produced by M-PARSER and Narmour could be due to any of the following causes:

- an inconsistency in Narmour's theory,

- a mistake by Narmour in applying the theory to the melody,
- the parser not fully applying the *Implication-Realisation Model* due to an intentional simplification or omission during the formalisation process,
- the parser not correctly applying the *Implication-Realisation Model* due to an unintentional omission or error in the formalised theory (and thus not having been encoded as part of M-PARSER),
- the parser not correctly applying the *Implication-Realisation Model* due to a coding mistake.

Since mismatches between M-PARSER and published analyses may be due to a number of possible inaccuracies in the encoding of Narmour's theory in M-PARSER, the experiments in this chapter also partially serve to test, and refine, M-PARSER. In each case where there is a difference between Narmour's published analysis and that produced by M-PARSER (or where a known inconsistency or simplification has meant that the formalised theory differs from the informal descriptions of Narmour's) we discuss which of the above possible reasons for the difference in analyses may apply.

Where there is no difference between the output of M-PARSER and Narmour's published analyses (and the aspects of the *Implication-Realisation Model* required to create the specific analysis do not contain any of those parts of the theory found to be unnecessary or which we chose to simplify during the formalisation process) we have assumed that this demonstrates that the aspects of the theory necessary to create the analysis have been correctly formalised and encoded. There is a however, always the possibility that part of the theory has been incorrectly formalised, and that a "cancelling" coding mistake has occurred, which just happens to result in the same analysis as a correctly formalised and coded theory would have generated. Further testing on large numbers of melodies would assist in tracking down such cancelling errors.

In the previous chapter we have described in detail the process by which M-PARSER hierarchically analyses melodies, identifying structures and points of closure. The testing has been carried out by comparison of analyses produced by M-PARSER with those published by Narmour for a set of test melodies. Some are excerpts of published musical works, others are synthetic melodies, presented by Narmour to illustrate various aspects of the *Implication-Realisation Model*. The next section describes the choice of test melodies, the remainder of the

chapter presents each test and discusses the results.

6.2. The choice of test melodies

Melodies from nine categories of published analyses were chosen, to comprehensively test all the basic features of the *Implication-Realisation Model*; however, parts of some of the melodies selected include aspects of analysis which Narmour's theory addresses (in his published analyses) but which are not described in any detail in Narmour's publications. Sections 6.2.2. to 6.2.9. describe the tests for melodies exemplifying the following parts of Narmour's analytical theory:

- **§6.3.** prospective structure,
- **§6.4.** retrospective structure,
- **§6.5.** monads and dyads,
- **§6.6.** closure,
- **§6.7.** hierarchy (in general),
- **§6.8.** non-systematic hierarchical promotion, and
- **§6.9.** chaining of structures.

The order of these tests is intended to roughly follow the order of the presentation of Narmour's theory in earlier chapters, which is generally a progression in complexity of parsing. It is only the subject of the tests for sections 6.6 and 6.7 that relate to the main simplifications made during the formalisation process, and the identified inconsistencies of Narmour's own applications. Therefore it is the tests for these parts of the *Implication-Realisation Model* where differences were thought likely, (and one was found in section 6.8.), between Narmour's hand-produced analyses and those generated by our computational parser.

In the following sections we present test pieces for each of the aspects of the theory in turn. The test pieces are presented in Common Music Notation and annotated with Narmour's published parses. For each in turn we indicate the results of the tests when run through M-PARSER. Where there was a difference between M-PARSER's analysis, and Narmour's published one, both are presented and the reason for the differences discussed. Where there was no difference between the two analyses the implications of the reproduction of Narmour's analysis by M-PARSER are discussed.

6.3. Tests for prospective structures

The melody fragments illustrated in Figure 6.1 are artificial melodies, produced by Narmour (1989)[1] to illustrate the simple forms of the eight prospective structures: P, IP, VP, R, IR, VR, D and ID.

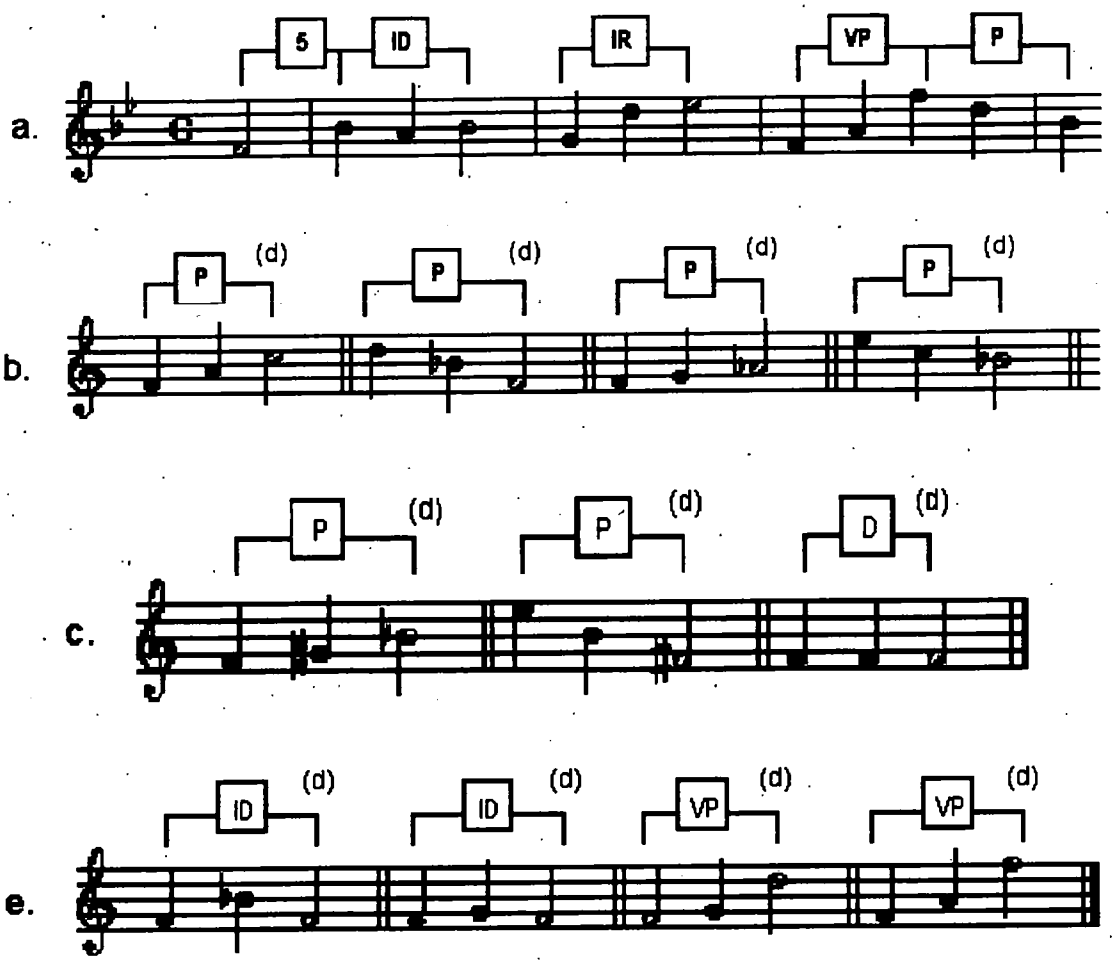

Figure 6.1: Melodies illustrating prospective structures.

Each melody fragment has been run through the parser. For each of the eight prospective structures the parser successfully identified the class.

[1] The melodies in Figure 5.2 are drawn from Narmour's publications as follows (his figure numbers and publication dates): (a) fig. 1.1a , 1990; (b) and (c) fig. 3, 1989; (d) and (e) fig. 4, 1989; (f) fig. 5, 1989.

For the case of prospective structures this confirms that the predicate-calculus descriptions in Chapter 4 are in accordance with Narmour's descriptions and application of his theory.

6.4. Tests for retrospective structures

The melodies in Figure 6.2 are artificial fragments presented by Narmour (1992) to illustrate seven of the eight possible retrospective classes of structure.

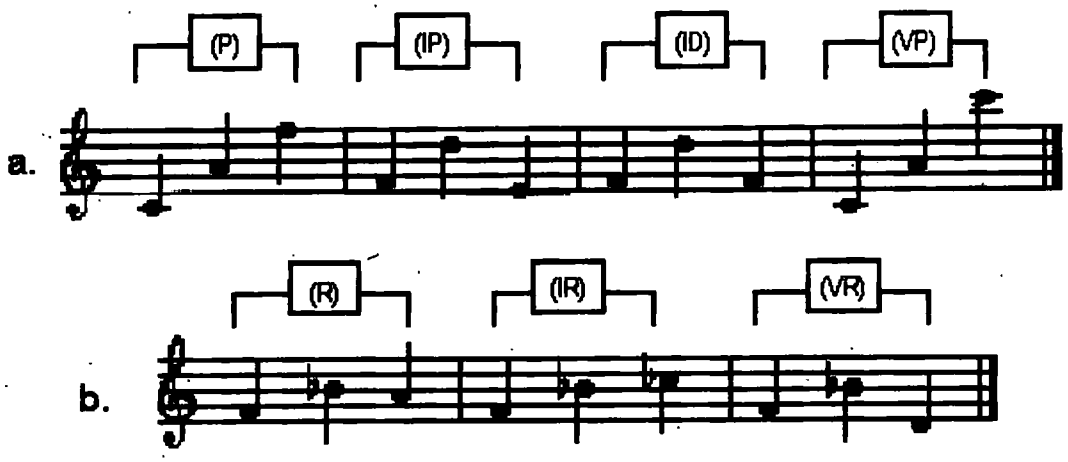

Figure 6.2: Melodies illustrating retrospective structures.

Each of the seven classes of retrospective structure above has been run through M-PARSER and been correctly identified. The eighth category of retrospective structure, retrospective duplication, cannot currently be detected by M-PARSER. This is related to the issue of how to model top-down stylistic closure, which has been discussed earlier and is also the basis of one of the further research projects described in Chapter 8.

6.5. Tests for monadic and dyadic structures

The melodies in Figure 6.3 (from Narmour, 1990) illustrate both dyads (at the musical surface), and monads at the second hierarchical level. These test pieces illustrate metric closure and accent closure, which are both features of Narmour's theory which he applies but does not attempt to describe.

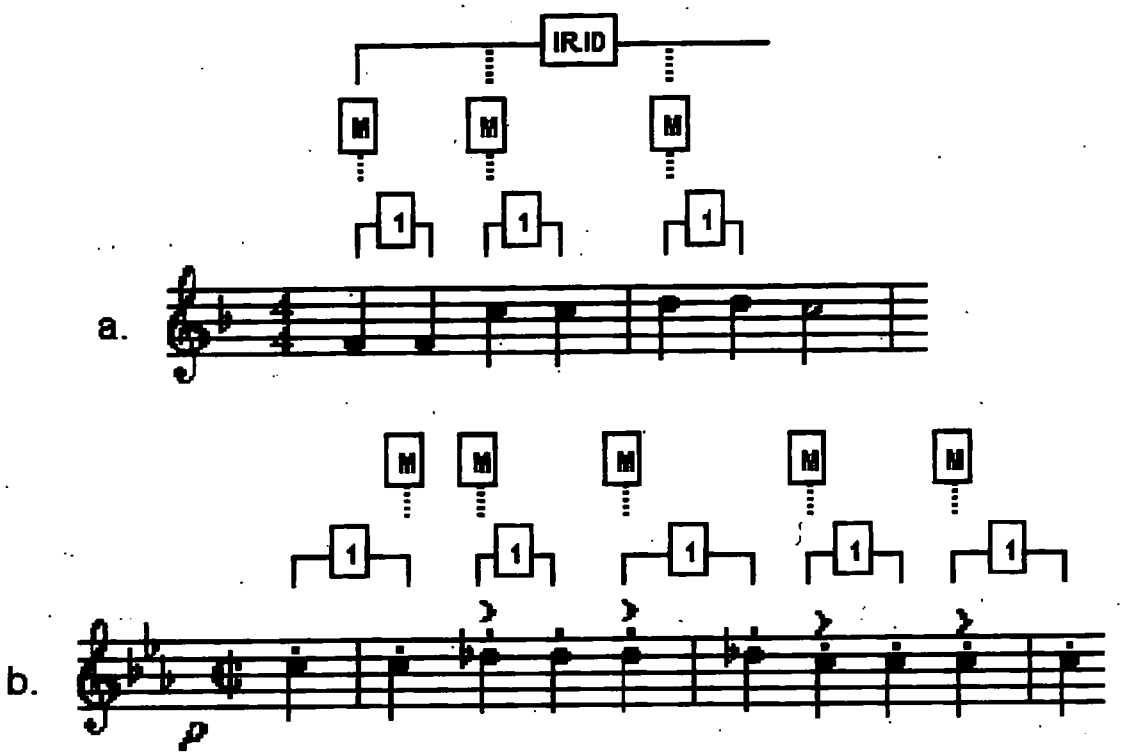

Figure 6.3: Melodies illustrating monads and dyads.

These melodies have been successfully parsed by M-PARSER. However, in both cases closure was modelled by its introduction into slots in the encoded melodies — metric closure for example (a) and accent closure in example (b). The insertion of values into slots for the parsing is essentially the hard-coding of reasonable metric and accent judgments by a musician, but which have not been described by Narmour. We have implemented a form of metric closure, which does differentiate between notes with different metric strengths, but currently it does always make the same distinctions about degree of difference of metric strength between two notes that Narmour makes. These tests have highlighted two aspects of the *Implication-Realisation Model* which Narmour has not described.

6.6. Tests for closure of structures

The melodies in Figure 6.4 (from Narmour, 1990) illustrate closure due to metre and duration.

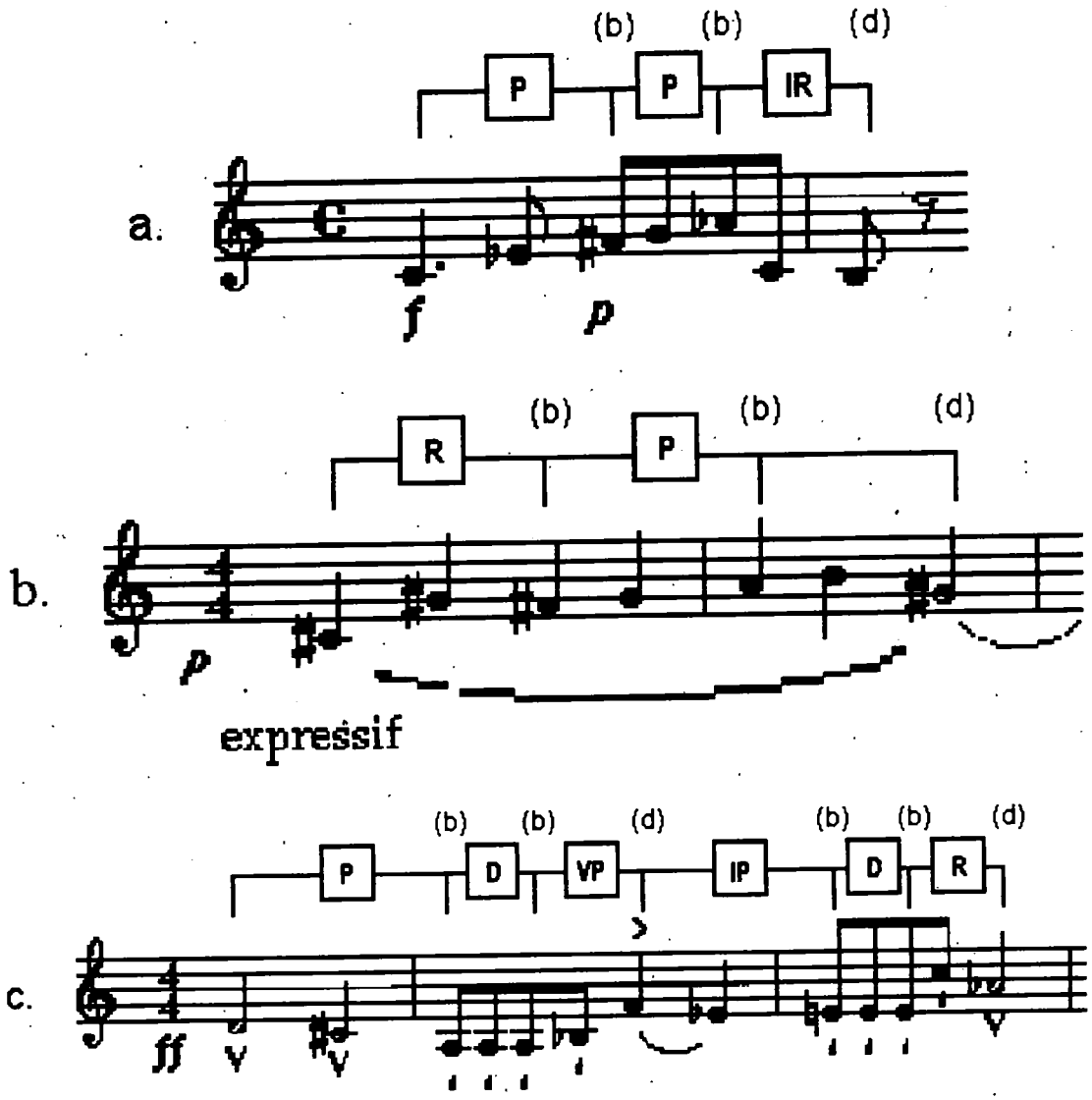

Figure 6.4: Melodies illustrating closure of structures.

Durational closure was correctly found in each instance.

As described above, metric differentiation is recognised by M-PARSER, but not always to the same degree as Narmour's analyses exhibit. In summary, on the quarter, half and three-quarters of the bar, share closure was usually found by M-PARSER, but the first note of each bar generally exhibited separate closure.

6.7. Tests for hierarchical promotion (in general)

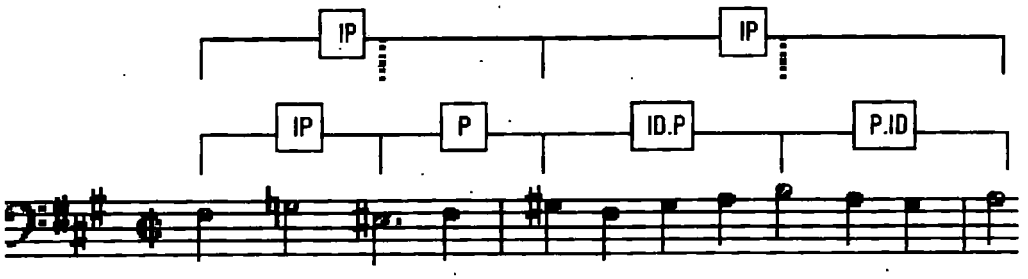

Figure 6.5: Melody illustrating hierarchical promotion.

When given slots to counter the metric closure problems outlined in the previous section, M-PARSER correctly promoted the notes for the melody in Figure 6.5 (and identified the two [IP] structures) — this figure is reproduced from Narmour 1989, Figure 8, p. 54: Bach, *Mass in B Minor,* Kyrie Eleison II, mm.1-3. This demonstrates that M-PARSER correctly promoted notes in the analysis hierarchy, given appropriate measures of closure.

6.8. Tests for non-systematic hierarchical promotion

The analysis in Figure 6.6. is from Narmour (1989) — Verdi, *I Lombardi,* Act IV, sc. 2. Such analyses are particularly interesting (as discussed in the previous chapter), because they demonstrate Narmour's non-systematic application of his theory, in that notes can "skip" levels when promoted. This is an example of an inconsistency in the way Narmour applies his own theory.

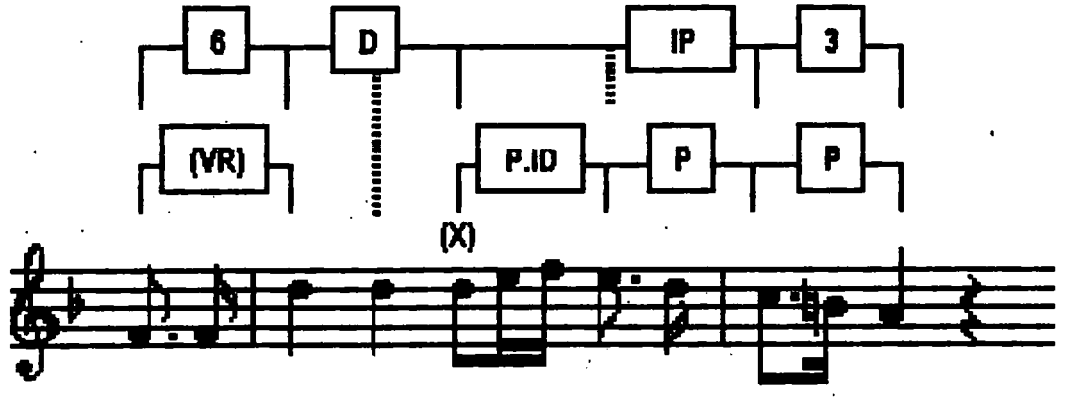

Figure 6.6: Example of a Narmour "skipped level" analysis.

It would appear that this aspect of level "skipping" could be modelled in a manner consistent with the theory, by the use of monads — as long as there was a sufficient measure of closure on the note to "skip" a level. In addition to being inconsistent, Narmour does not provide description of the criteria for notes that should skip a level, therefore this is also an example of another gap in his description of the *Implication-Realisation Model.*

M-PARSER was able to reproduce the top level of the analysis shown in Figure 6.6 by the use of monads, and closure statements in the slots of the encoded melody. The closure for the monad note had to be placed in a slot in the melody, since although M-PARSER identified some closure due to metre, the influence of extra-opus style (indicated by the "(X)" in the analysis) is one of the features not described by Narmour. The M-PARSER analysis for the melody is shown in Figure 6.7.

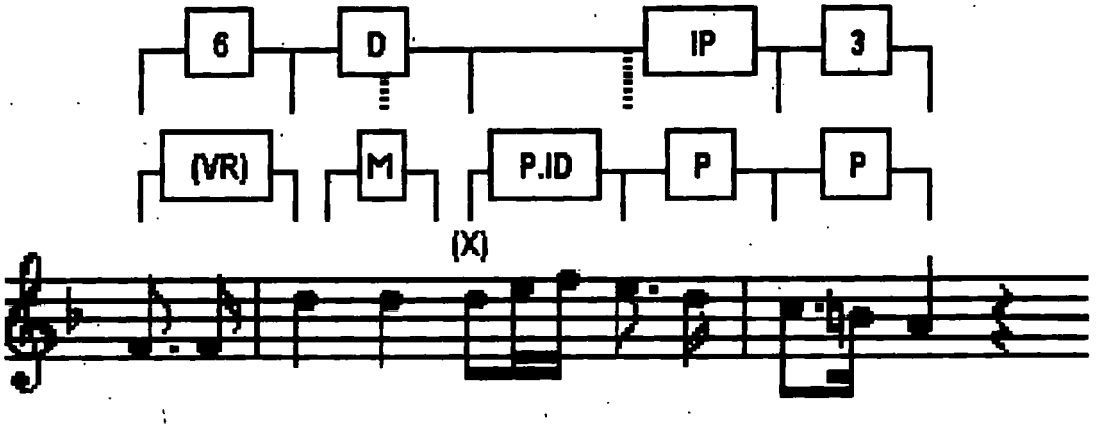

Figure 6.7: M-PARSER solution to avoid "skipping" levels.

6.9. Tests for chaining (and combination) of structures

The test melody shown in Figure 6.5 was also used to demonstrate the successful modelling of chains by M-PARSER. The chained structures of **[ID.P]** and **[P.ID]** were successfully parsed (with appropriate closure slots in the melody). Note that chains (a merger of three or more structures) and combinations (Narmour's distinction for mergers of just two structures) are modelled in the same way (i.e. 2 or more is a chain). The melody serves to demonstrate parsing of both chains and combinations (MELODY-ED has been successfully tested with melodies exhibiting chains of 3 and 4 structures).

6.10. Analysis of results

We can categorise and account for differences between the M-PARSER analyses and Narmour's analyses as follows.

An inconsistency in Narmour's theory

We would propose that Narmour's non-systematic application of hierarchical promotion (discussed in section 6.8) is inconsistent with the goal of having an explicit, "scientific" theory. If there are circumstances when such jumping of levels can occur the theory should be re-worked such that it can explain such jumping in a consistent way. However, we cannot claim to have discovered this inconsistency through testing, since Narmour draws attention to his inconsistent use of his theory himself (Narmour 1990, p. 183).

A mistake by Narmour in applying the theory to the melody

We found no evidence from Narmour's manual analyses that did not agree with his verbally described theory.

The parser not correctly applying the Implication-Realisation Model due to an intentional simplification or omission in the formalised theory (and thus not having been encoded as part of M-PARSER)

All closure due to top-down style comes under this heading, such as recognition of all cases of retrospective duplication, and the influence of style on any other structure. These aspects of the theory have been intentionally omitted from the formalised theory. See Chapter 8 for discussion of how they might be modelled.

Closure due to harmony and metre have been formalised in the previous chapter, and as demonstrated in the results presented in this chapter, metric closure has some degree of "fine tuning" to be done to it (harmonic closure suffered similar problems, but due to the simplistic nature of our implementation of the closure, fine tuning will be insufficient — see the further work section of Chapter 8 for our proposals for a more sophisticated harmonic analysis procedure).

The parser not correctly applying the Implication-Realisation Model due to an unintentional omission or error made when encoding the theory

The kinds of "fine tuning" of closure strengths referred to above formed an informal correction of unintentional oversimplifications. No other unintentional omissions or errors in the formalisation were found during the testing at this stage.

The parser not correctly applying the Implication-Realisation Model due to a coding mistake

While iteratively coding, mistakes were identified through the use of the test melodies in this chapter, and other synthetic melodies. However, all the mistakes were corrected before the collection of the test results we have just presented.

6.11. Conclusions

All differences between M-PARSER outputs and Narmour's published analyses can be explained in terms of: (1) simplifications made when the theory was encoded in M-PARSER; (2) those aspects (such as style) which have not been encoded at all due to the lack of a description of these aspects of the theory by Narmour; and (3) the highlighted case of Narmour's non-systematic promotion of notes in the hierarchy. However, the simplifications were for such concepts as harmony (consonance and dissonance) and metre, and had to be made since these aspects of the *Implication-Realisation Model* are not fully laid out in Narmour's writings. Indeed, harmonic analysis is a topic of considerable research in artificial intelligence and music, and to date only limited success has been achieved in this complex (and ill-defined) task.

M-PARSER has correctly identified all classes of prospective structure, seven of the eight retrospective structures (not the eighth due to the lack of definition of stylistic closure). The parser has successfully recognised durational closure, hierarchically promoted notes, and created chain structures when given appropriate measures of closure. By the identification, and explanation, of differences between the analyses from M-PARSER and Narmour's (informally applied) analyses, we have highlighted those aspects of M-PARSER, and Narmour's theory, that require further formalisation, when such aspects of the theory are published by Narmour. These aspects of the theory can be summarised as follows:

- closure due to style (not currently implemented due to the lack of

any description by Narmour, and the size of the research project involved),

- metric closure (our formalism needs work to fit with Narmour's analyses, however, we have demonstrated that the measure of metric strength adopted works, but not quite in the same way Narmour applies his unexplained measure of metre),
- harmonic closure (our simplified measure of harmonic closure works, but not in the way Narmour applies his own, undescribed measure of harmonic closure),
- accent closure (not described by Narmour, and not currently modelled in our clarified theory),
- hierarchy skipping (both the inconsistency in Narmour's application of his own theory, and the gap in his theory due to the lack of description of when skipping should occur have been identified through the testing procedure).

We have demonstrated that M-PARSER is a full implementation of both the primitives of Narmour's theory (given the simplifications and omissions noted above), and of the procedural method for the note-by-note application of the theory to a melody at many hierarchical levels simultaneously. The computational modelling of our clarified theory both supports formalisation we have proposed, and has produced an explicit, unambiguous version of Narmour's theory in the form of the computer program.

Chapter 7:
MOTIVE: A tool for constraint-based generation of melodies

"A WORKING DEFINITION OF CREATIVITY

...

Second, the process has no precise goal, but only some pre-existing constraints or criteria that it must meet." (Johnson-Laird 1988, p. 255)

"... the theorist's work is not done until analysis leads back to synthesis."
(Narmour 1984, p. 197)

7.1. Introduction

The long term aim of our research is to work towards the development of an intelligent learning environment for novice composers of melody. So far in this thesis we have described a process of formalising and computationally modelling Narmour's analytical theory for melody. In this chapter we aim to answer the question of how one might use such a computational instantiation of the *Implication-Realisation Model* educationally.

We present the chapter as follows:

§7.2 a brief description of a hypothetical educational scenario,

§7.3 detailed description of the design and implementation of MOTIVE a constraint-based tool for the generation of melodies, and

§7.4 a summary of the limitations of the current prototype.

The scenario we present is intended to result in educational benefits for novice composers of melody — we discuss what the educational benefits might be expected to be, but the determination of whether the stated benefits actually result from such learning experiences requires future empirical investigation (see the further work section of Chapter 8).

7.2. A hypothetical educational scenario

7.2.1. Melody generation by metric and harmonic constraints

Melodies can be generated by applying harmonic and metric constraints to notes occurring in particular levels of a hierarchical analysis. Such constraints might be very restricting at high levels, and weakened for each lower level. For example, harmonic constraints can take the form of movement through Lerdahl's (1988) "Pitch Spaces", a set of spaces which contain pitch classes of increasing harmonic importance (as reviewed in Chapter 2). There is also the built-in constraint that a note appearing in a particular hierarchical level must generate sufficient closure in the next lower level, to justify its place in the hierarchy.

Described below are metric and harmonic constraints for an example of a constrained melody generation. The example of the hierarchical generation itself is shown graphically in Figure 7.4 (a few pages ahead). The example can be thought of as both a bottom-up, analysis of the melody, and a top-down, hierarchical construction. If considered as an analysis, the melody is the data, and the artefact created is the analysis (a set of Narmour structures). When considered as a top-down generation, the artefact would be a constraint network, and a family of generated melodies meeting the constraints. The difference being that analyses for existing melodies would not have explicit constraints for the hierarchical levels[1]. The example in Figure 7.1 is based on the following theme from Tchaikovsky's (1877) *Swan Lake*, illustrated in Figure 7.1 below.

[1] See Chapter 8 for a discussion of further work, including the design of a constraint inference module, during bottom-up analysis.

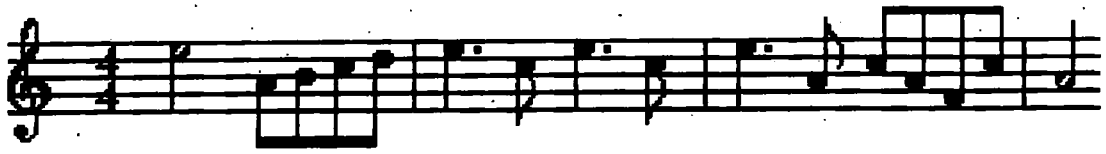

Figure 7.1: The Swan Lake Theme.

Before we present the generation from the scenario, the hierarchical application of metric and harmonic constraints are individually described. In addition to the hierarchical metric and harmonic constraints, initial decisions for the generation were a time signature of 4/4, and the key of A minor.

7.2.2. The Metric Constraints

The metric constraints in Figure 7.2 below show how the onset times of notes were highly constrained at the higher levels: H4, H3 only notes on bar boundaries, then half bar boundaries are allowed, where H4 is the highest level, and H1 the lowest, and the order of note generation is H4, H3, H2, H1. As one moves down the hierarchy the constraints are relaxed (so that at levels H1 and H2, notes are allowed on eighth bar boundaries).

```
 (Time signature =  2/4)
H4 : notes on bar boundaries
     | O              | O              | O              |
H3 : notes on half bar boundaries
     | O        O     | O        O     | O        O     |
H1 and H2 :     notes on eight bar boundaries
     | OOOO     OOOO  | OOOO     OOOO  | OOOO     OOOO  |
```

Figure 7.2: Metric constraints used for Swan Lake generation.

7.2.3. The Harmonic Constraints

The harmonic constraints (shown in Figure 7.3) are relaxed in a similar way as one moves down the hierarchy. Initially (H4) only notes in *Open Fifth Space* are allowed (the second highest of Lerdahl's pitch spaces). At each level down, notes from less consonant pitch spaces are permitted, ending with any non-chromatic note (*Diatonic Space*) being allowed for the lowest levels (H1 and H2).

(Key = Am)

H4 : [A, E]

H3 : [A, C, E]

H1, H2 : [A, B, C, D, E, F, G]

Figure 7.3: Harmonic constraints for the Swan Lake generation.

The lowest two levels of the constrained generation (H1 and H2) are closely linked. H1 can be thought of as a correcting level, where notes (such as the two E's in bar 2) which did not have sufficient closure for hierarchical promotion are given extra closure by the shortening of following notes (i.e. the shortening of the two C's in bar 2). In this way any over-simplifications due to hierarchical constraints are countered, resulting in more correct Narmour analyses for generated melodies.

Figure 7.4 presents the ***constrained generation*** (i.e. the notes in higher levels were generated first, each subsequent level being an "embellishment" of its parent level). As can be seen in the figure the lowest level generated (level H1) reproduces the Swan Lake theme. However, many other, related melodies are also generated that fit the same constraint network, as we now explore.

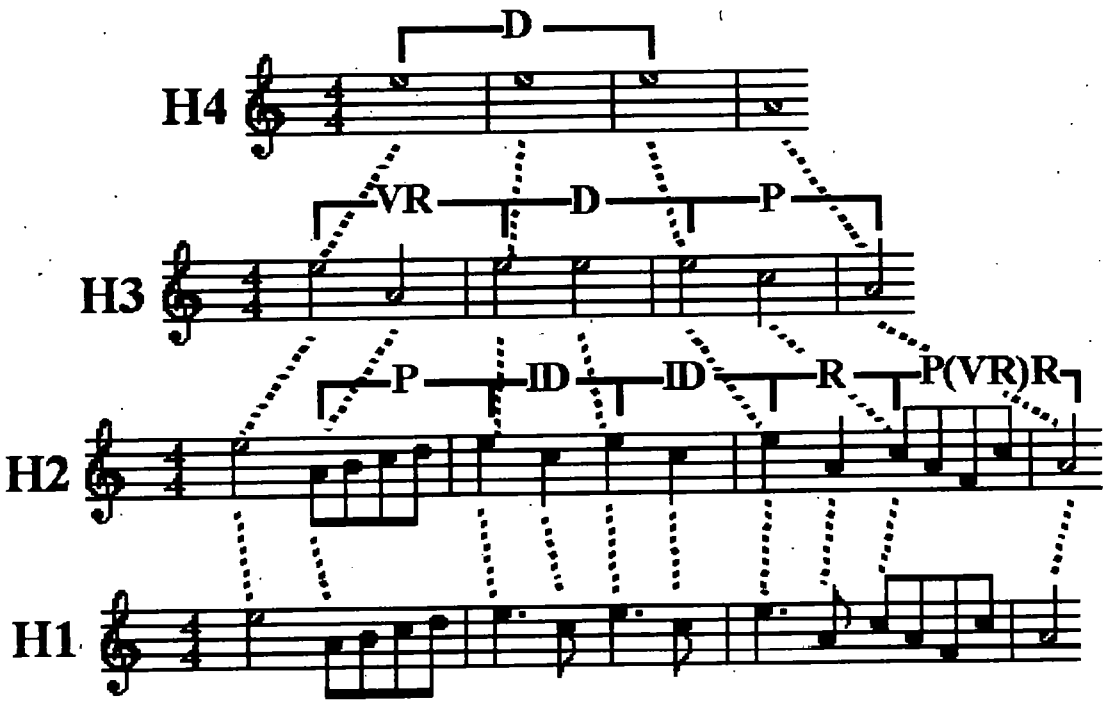

Figure 7.4: MOTIVE constrained generation of Swan Lake theme.

Figure 7.5 below presents one alternative generation, which meets all the same metric, harmonic and Narmour structure constraints, but where levels H2 and H1 are different.

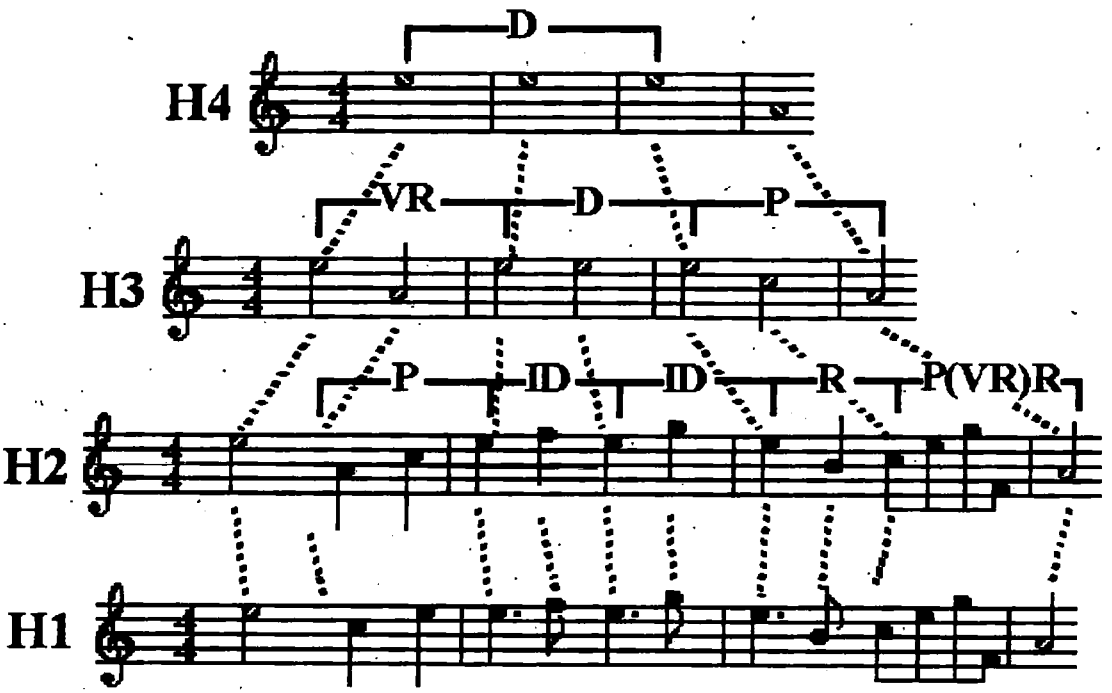

Figure 7.5: MOTIVE generation with levels H2 and H1 different from original.

Figure 7.6 below presents another alternative, where an alternative generation for level H3 is chosen — note that at level H3 only variations in the VR structure are possible, due mainly to the strict harmonic and Narmour structure constraints. At the higher levels, when more restrictive constraints are applied, few alternatives are possible, whereas at the lower levels many more variations are possible. To date we have only experimented with constraint hierarchies of up to 4 levels, however, were hierarchies of more than this number of levels to be used then many more generations would be possible. This can lead to problems of the user being overwhelmed by too many alternatives, a problem that is explored at the end of this chapter.

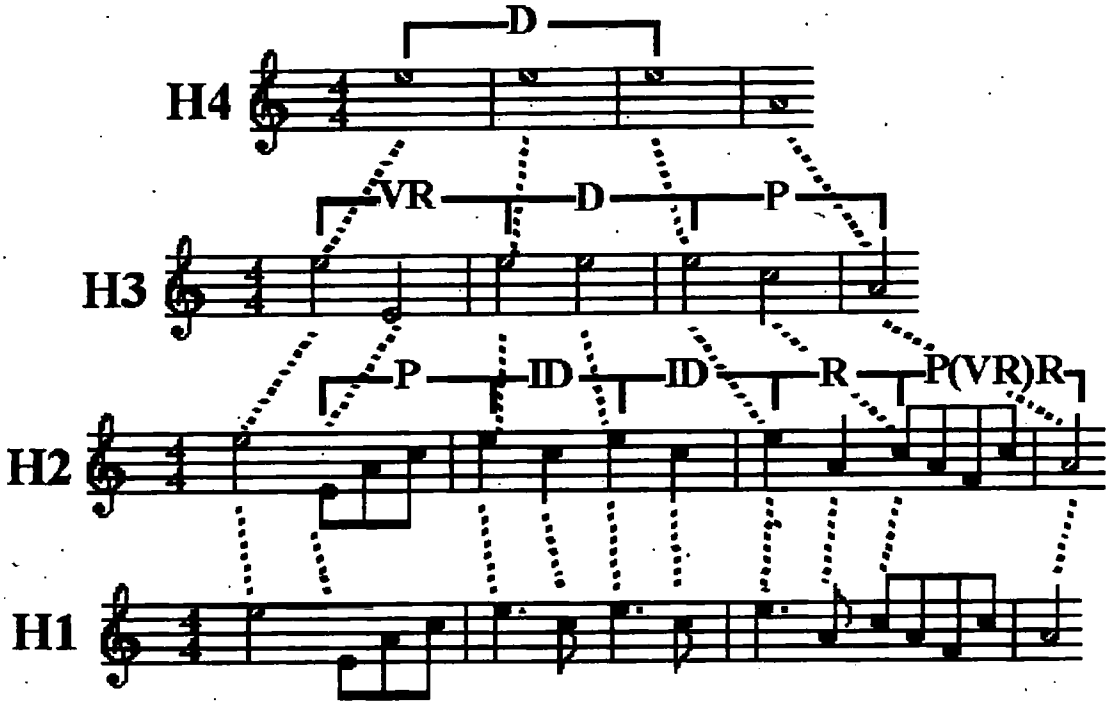

Figure 7.5: Generation with levels H3, H2 and H1 different from original.

7.2.4. Discussion and analysis of the scenario

The various educational aspects of the scenario described above

The scenario illustrates how a student can be given a composition task as an initial set of constraints (for example the harmonic and metric context, a number of bars for the melody and a pitch range within which to generate a melody). The student can use such a constraint-based tool to move between different hierarchical levels, try out constraints by defining them and running the generator, and then criticise the results leading to changes in the constraints and regeneration in an iterative approach to analysis and composition.

For the purpose of designing a framework for educational systems for melody composition we will assume that such learning experiences have the following educational benefits:

- use of the tool helps develop a constraint-based view of composition,
- students learn to apply variations to "interesting" fragments of melody,
- students work at both the local and global level when composing, and
- by having to apply and work within harmonic, metric and phrase variation constraints students develop a better understanding of these features of music and

their relation to composition.

Requirements of intelligent learning environments able to support the scenario

An intelligent learning environment would require a number of system components to make possible the scenario presented in section 7.2.1. They can be summarised as follows:

- a constraint-generator (taking a set of constraints on variables, and variable domains, as input, and outputting sets of variable assignments for which all the constraints are satisfied, or if the constraints cannot be satisfied, outputting those constraints and variables that are in conflict),
- a knowledge base of musical constraints,
- a corpus of musical examples, from which melodies can be chosen for analysis and as the starting point for a new melody (for example via the "Paul Simon" composition method of repeatedly making changes to an existing piece of music to create a new composition — see section 2.4.3. of Chapter 2),
- a "toolbox" of musical tools, containing at least one tool able to perform analyses, and one to perform generation (both in terms of the *Implication-Realisation Model*) — the tools would probably require a direct manipulation front end to "shield" the student from the technical details of the constraint satisfaction process (see next chapter for details),
- a tutor to set initial tasks for the student, and guide the student in the use of the constraint-based tools.

It is with these system components in mind that in the further work section of Chapter 8 we present MELODY-ED, a framework for an intelligent learning environment able to support such educational scenarios as that presented in section 7.2.1.

7.3. A constraint-based tool for melody

This section describes constraint-based extensions to the parser (M-PARSER) presented in Chapter 5. These extensions to M-PARSER allow the generation of melodies that conform to given constraints. The constraint-based extensions are called M-CONSTRAINT, and together

M-CONSTRAINT and M-PARSER form the melody generation tool called MOTIVE.

We will describe M-CONSTRAINT from the following viewpoints:

- the AI technique called "generate and test" (e.g. Winston, 1981), with reference to how it can be applied to generating melodies given an analysis based on the *Implication-Realisation Model*, and
- additional types of constraints, for reducing the set of generated melodies sensibly in terms of metric and harmonic constraints.

We describe the properties of the "family" of melodies each set of constraints can generate, and presents an example of a melody generated using the techniques described. This section is followed by one that critically summarises the constraint-based methods presented in this section, both in terms of limitations due to technique, and limitations from an educational and musical point of view.

7.3.1. Generate and test constraint-satisfaction

The AI technique of "generate and test" (e.g. Winston, 1981) can be used with left-to-right, top-to-bottom backtracking as a technique for constraint-based generation. This technique has been explored and criticised for some time (for example Guesgen and Hertzberg, 1991). In general a constraint-based problem can be formally expressed as a set of variables, a set of corresponding domains for the variables, and a set of relationships between the variables. A solution is expressed as a statement of a value for each variable from its corresponding domain such that all the stated relationships between the variables are true.

The stages of "generate and test" as applied to the constraint-satisfaction process are as follows:

(1) generate a candidate solution — generate a possible solution by assigning each variable a value from its corresponding domain, in a combination that has not already been put forward as a candidate solution,

(2) test the candidate solution against the variable relationships — test the relationships between variables as defined in the statement of the problem,

(3a) test if solution has been found — if all the relationships are true, a solution

has been found,

(3b) if solution not found generate next solution — if one or more of the relationships is not true, backtrack to stage (1) (i.e. try a new set of variable assignments) — if stage (1) cannot be redone then all possible combinations of variable assignments must have been tried and the problem is not solvable for the given domains and variable relationships.

In the "generate and test" procedure naive backtracking is used to automate the choice of variable assignments from their corresponding domains to systematically test all possible candidate solutions.

In general, given a manageable, finite domain, simple backtracking can be used. For simplicity, this is the technique we have used in MOTIVE.

7.3.2. Generate and test applied to Implication-Realisation Model analyses

Since the output of constraint-based generation from an *Implication-Realisation Model* analysis is a melody, the first step is to define the variables, and their domains, associated with a melody. The following is a summary of how MOTIVE represents a melody (more detail can be found in Chapter 5 and Appendix D).

A melody is represented as:

- a list of notes (each note having the variables of pitch, onset time and duration),
- a list of the metres associated with each part of the melody,
- a list of the harmonic scales associated with each part of the melody,
- a list of the chords associated with each part of the melody.

Domains for each of the variables are defined as follows (the ordering of the following lists of domains can improve efficiency, and is discussed in section 7.3.5):

- **note pitch** [C3, C#3, Db3, D3, ..., B4, C5][2],

2 A range of two octaves has been chosen for two reasons: first, most melodies fit within a range of two octaves; second, such a range restricts the number of possible note pitches to under 50. This small range of possible pitches is important since it means the constraint-generation can be carried out fast enough to support real-time, visible changes (a requirement of direct manipulation interfaces). Of

- **note duration** [demisemiquaver, dotted-demisemiquaver, double-dotted-demisemiquaver, semiquaver, dotted-semiquaver, double-dotted-semiquaver, quaver, dotted-quaver, double-dotted-quaver, crotchet, dotted-crotchet, double-dotted-crotchet, minim, dotted-minim, double-dotted-minim, semibreve, dotted-semibreve, double-dotted-semibreve],
- **note onset time** the beginning of the first bar plus any multiple of the smallest note durations that have no factors but themselves — i.e. 0 + (N1 * demisemiquaver) + (N2 * dotted-demisemiquaver) + (N3 * double-dotted-demisemiquaver), where N1, N2, and N3 are integers and the durations of notes are expressed in MTUs (see Chapter 4 and Appendix C),
- **metre list** a list of elements, where each element is an ordered pair of values, the first being a valid note onset time and the second being a metric vector (see Chapter 4 and Appendix B) in the following form: [[Numerator, Denominator], List_of_beat_subdivisions, Bar_duration].
- **scale / key list** a list of elements, where each element is an ordered pair of values, the first being a valid note onset time and the second being a scale — scales are represented as an ordered pair of values

  [Root, Scale_type]

  Root — root pitch class (Ab .. G#),

  Scale_type — a member of the list [major, harmonic_minor, natural_minor]
- **chord list** a list of elements, where each element is an ordered pair of values, the first being a valid note onset time and the second being a chord — chords are represented as an ordered pair of values

  [Root, Chord_type]

  Root — root pitch class (Ab .. G#),

  Chord_type — a member of the list [major, minor, augmented, suspended, major_seventh, minor_seventh, dom_seventh, full_diminshed_seventh, half_diminshed_seventh, ninth]

course, there is no problem in principle for having an arbitrary range for note pitches. In addition, the "family" of melodies generated could be modified by changing size of domain, or the starting pitch.

7.3.3. An example of a melody generation problem

Before looking at the issues involved in adding educationally and musically sensible constraint, we present a simple constraint-based melody problem, in order to give the reader a feel for what is involved.

A problem to be solved could be expressed by the following sentence:

> *"generate a 2-bar melody fragment which would result in the analysis illustrated in Figure 7.7"*

Such a problem could arise, for example, in an educational context in which a student has been studying well known melodies containing phrases that conform to this pattern, and then was challenged to produce a similar melody using MOTIVE.

In an educational context, such problems would not be presented in a technical form, but via a direct manipulation interface, such as that presented in the further research section of Chapter 8. The remainder of the focus of this section is not on the educational aspects, but on the technical details of constraint generation.

Figure 7.7: The analysis on which to base a generated melody fragment.

The [P] and [D] refer to the process and duplication structures described in Chapters 3 and 4. The process is a sequence of small intervals either all ascending or descending. The duplication structure is a sequence of repeated notes. The "(d)" symbolises closure due to long durations.

We can first infer a number of constraints from the analysis in Figure 7.7 as follows (a number of the constraints take into account that M-CONSTRAINT represents a rest as a type of note, with no pitch, but having duration):

- the melody fragment will be made up of four groups of notes: any rests at the beginning of the first bar, the notes of the first structure [P], any rests between the notes of the two structures, and the notes of the second structure [D],
- since the first structure [P] is not a dyad or monad, it must contain at least 3 notes,

- since the second structure **[D]** is not a dyad or monad it must contain at least 3 notes,
- since the two structures share no notes, the first structure can be generated without consideration of the second structure (and the second only need consider constraints on the onset time of its first note, in relation to the end of the last note of the first structure),
- the onset time of all but the first note of each structure must be at the point in time when the preceding note of the structure ends,
- the onset time of the first note of the second structure **[D]** must be at the point in time that the final note of the first structure **[P]** stops.

Since the analysis contains only one level of analysis, the promoted notes of each structure need have no implied relationship between them.

Thus the problem can be expressed more explicitly as follows:

- the notes of the melody fragment are a list (in temporal order) as follows: [AR1, AR2, ..., ARn, P1, P2, ..., Pn, MR1, MR2, ..., MRn, D1, D2, ..., Dn], where the notes AR1..n are any initial (anacrusis) rests in the first bar[3], P1..n are the notes of the process **[P]** structure, MR1..n are the notes of any middle rests between the two structures, and D1..n are the notes of the duration **[D]** structure,
- three or more notes form a process structure (so: [P1, P2, ..., Pn], where $n \geq 3$),
- durational closure acts to close the process structure (therefore the duration of note Pn must be significantly, according to the formalised Implication-Realisation Model, longer than that of note Pn-1),
- three or more notes form a duration structure (so: [D1, D2, ..., Dn], where $n \geq 3$),
- durational closure acts to close the duration structure (therefore the duration of note Dn must be significantly, according to the formalised Implication-Realisation Model, longer than that of note Dn-1),

[3] Anacrusis rests are necessary because in M-CONSTRAINT the first note of every melody starts on the first beat of bar 1.

- the onset time of each note of all but the first note of the melody is at the point when the preceding notes duration ends, so:

  [N1, N2, ..., Nn] (all notes of melody)

  where $1 < m \leq n$,

  onset_time(Nm) = onset_time(Nm-1) + duration(Nm-1).

The following simplifying assumptions have been made:

(1) the first note of the process structure is the first note of the melody fragment (i.e. there is no anacrusis),

(2) only durations of minims and crotchets will be used, and

(3) the metre for the entire melody will be 4/4 time.

For our example below we have allowed notes to take any pitch (including chromatics), it is equally simple to constrain the generation to, say, only diatonic pitches.

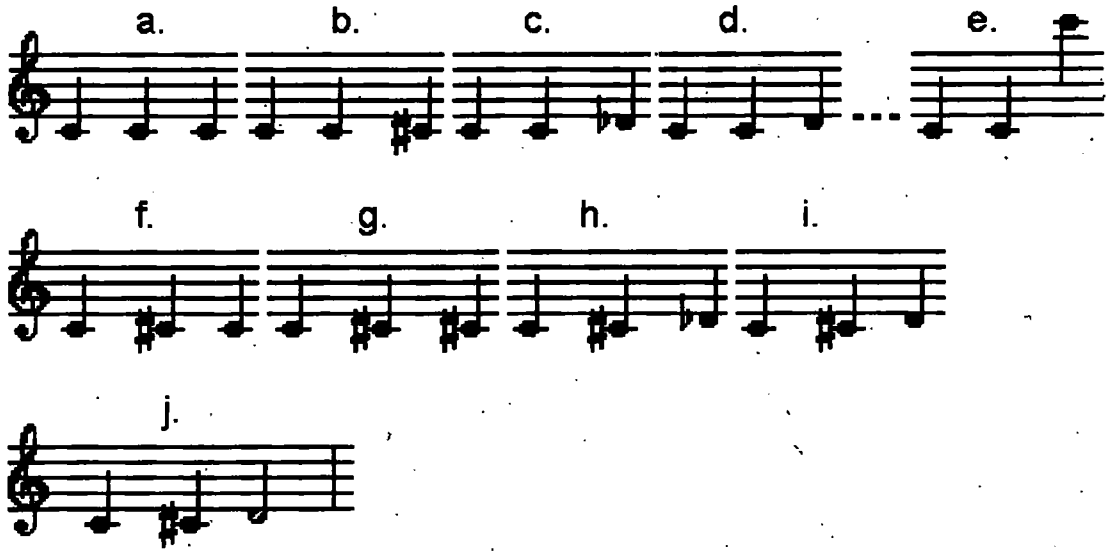

Figure 7.8: Sequence of melody fragments generated by simple backtracking, to fit first part of analysis from Figure 7.7.

Part (j) of Figure 7.8 shows the final three note melody fragment generated to fulfill the process structure of the analysis given in Figure 7.7 (the three notes of the melody fragment will be referred to as P1, P2 and P3 respectively). The first line of three notes melody fragments in Figure 7.8 (fragments (a) to (e)) show how different values for the third note (P3) are tried

because the pitch constraints of a process structure are not satisfied for the notes generated. The dots between parts (d) and (e) of the figure indicate all the intermediate pitch values for P3. Since the pitches for P1 and P2 are the same, no matter what value P3 has, the pitch constraints for a process structure will not be satisfied.

The second line of Figure 7.8 (parts (f) to (i)) illustrate the sequence of different values for note P3, once note P2 has been given a different pitch (i.e. C#). The three note melody fragment in part (i) satisfies the pitch constraints for a process structure, but does not satisfy the constraint that note P3 must have durational closure.

The final three note melody fragment of Figure 7.6 (part (j)), illustrates how note P3 is instantiated with a different duration (of a semibreve), and since the difference in duration between the minim of P2 and the semibreve of P3 is significant (according to the *Implication-Realisation Model*, see Chapter 3), note P3 has durational closure and a three note melody fragment to fit the process structure of Figure 7.7 has been successfully generated.

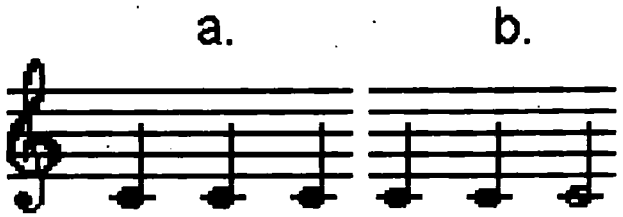

Figure 7.9: Sequence of melody fragments generated by naive backtracking, to fit second part of analysis from Figure 7.7.

Figure 7.9 illustrates the shorter sequence of three note melody fragments generated to fit the constraints for the second, duplication, structure of Figure 7.7 — the three notes of these melody fragments will be referred to as D1, D2, and D3 respectively. Although the three notes of part (a) of Figure 7.9 fit the pitch constraints for a duplication structure (i.e. they are all the same pitch), the duration of note D3 does not cause durational closure. As with P3 for the process structure, the new duration of a semibreve for D3 in part (b) of Figure 7.9 satisfies the constraint for durational closure, and the three notes for the second structure of Figure 7.7 have been successfully generated. The complete melody fragment is presented in Figure 7.10 below.

The melody fragment shown in Figure 7.9 is the *first* six note fragment generated to fit our example exercise. There are hundreds of different six note fragments that would also satisfy the

given simple constraints. In more complex examples, such as the Swan Lake example earlier in this chapter, as the number of notes and constraints increase, and become interrelated at different hierarchical levels, the number of possible solutions goes down. At the end of this chapter, and in Chapter 8, we discuss problems associated with large numbers of possible solutions that can occur in this kind of constraint-based approach.

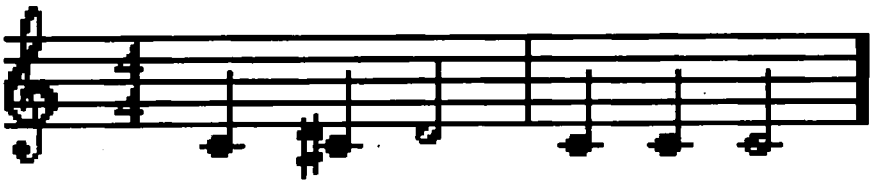

Figure 7.10: The generated melody fragment to fit the analysis from Figure 7.7.

Implicit hierarchical constraints

Since M-CONSTRAINT has been designed to generate constraints hierarchically, to correspond to an *Implication-Realisation Model* analysis — i.e. each note for each level of an M-CONSTRAINT hierarchy should directly correspond to a note in an analysis by M-PARSER. As described in Chapter 3, notes are promoted if they are an initial or terminal note of a structure (or of a combination or chain of structures according to strength of closure). This leads to an implicit constraint in the constraint-based hierarchy used for generation, such that any note appearing at a given level in the hierarchy (except the lowest level of the musical surface) must appear in all lower levels in the hierarchy. In addition, at each lower level the note must have sufficient closure (through being an initial or terminal structure note) to justify its place in each higher level.

7.3.4. Three additional types of constraints

Metric (and consequent durational) constraints

Hierarchical metric constraints can be generated from the representation of the metre(s) for a melody (see Chapter 5 and Appendix D for a detailed description of how M-PARSER represents the metre(s) of a melody). Most metres break down a bar into two or three beats, and then those beats into two or three sub-beats and so on — the factor for each hierarchical subdivision is usually either 2 or 3. The metric vector representation for such metres is a list of integers representing these factors. From these factors a hierarchy of metric constraints can be generated, each level in the hierarchy being a relaxation of the beats allowed in the previous

level, by allowing notes on sub-beats according to the next factor in the metric vector.

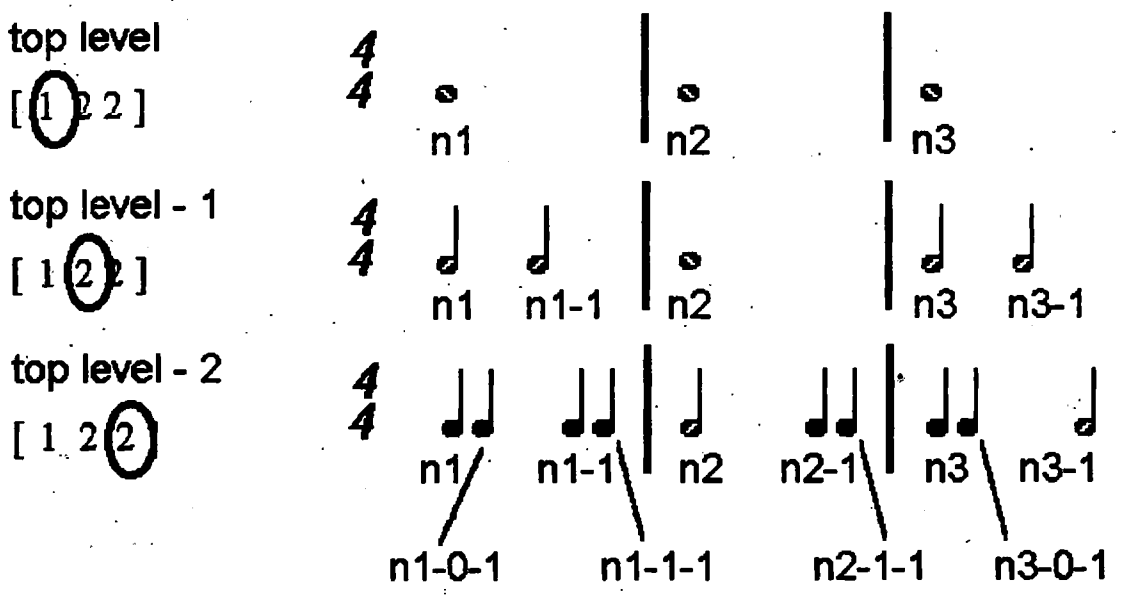

Figure 7.11 Hierarchical metric constraints for metric vector [1, 2, 2].

Figure 7.11 illustrates the top three levels of hierarchical metric constraints from the metric vector `[ 1, 2, 2 ]` — this metric vector is a representation of common time (4/4). Currently the factor list for all metric vectors start with a one, representing that the beat at the beginning of a bar is the most important. The third argument of an element in the metre list for a melody states the duration of a bar (in the case of the metre for Figure 7.11, a semibreve). The metric constraint for the top level of Figure 7.11 is that notes must have onset times of the beginning of a bar, and the duration of notes must be an integer number of bars (e.g. notes "n1", "n2" and "n3" in Figure 7.11).

The second highest level of Figure 7.11 ("top level - 1") illustrates how the metric constraint has been relaxed to allow additional notes to be added to the melody, whose onset times are on half bar beats — this relaxation is derived from the factor 2, being the second element in the list of factors for the metric vector. The corresponding durational constraint for this level is that additional notes in this level, and the notes at the beginning of the effected bars, must have durations of half a bar (i.e. a minim for this metre). Thus the onset times of existing notes "n1", "n2" and "n3" do not change, but the durations of "n1" and "n3" are halved to allow for the insertion of notes "n1-1" and "n3-1". Bar two of this level illustrates that although the metric constraint has been relaxed, extra notes do not have to be inserted into every position now

allowed by the relaxed constraint.

The third from top hierarchical level in Figure 7.11 ("top level - 2) illustrates how the third factor of the metric vector (another 2), further relaxed the metric constraint to allow notes with onset times of a quarter of a bar. Once again, the corresponding durational constraint is that for inserted notes, and those notes that immediately precede the inserted notes, must have a duration of a quarter of a bar (i.e. a crotchet for this metre).

Some metres sub-divide bars by factors that are not 2 or 3, for example other primes such as 5 and 7. For any prime factors other than 1, 2 and 3 the odd number of sub-beats may be grouped (for example a bar with 5 beats may be grouped 3 beats then 2, the first and fourth beats of the bar being more important than the others). As described in Chapter 5 and Appendix D, when such a sub-division of a bar occurs it is represented in a metric vector by a list of the note groupings, thus a metric vector element of `[ 1, [3, 2] ]` represents a bar with 5 beats, grouped into 3 then 2. Each such beat grouping corresponds to two metric constraints, the first for the implied important beats at the beginning of each group, and the second for how many beats follow the first in each group. Such a metre and its corresponding metric and durational constraints is illustrated in Figure 7.12.

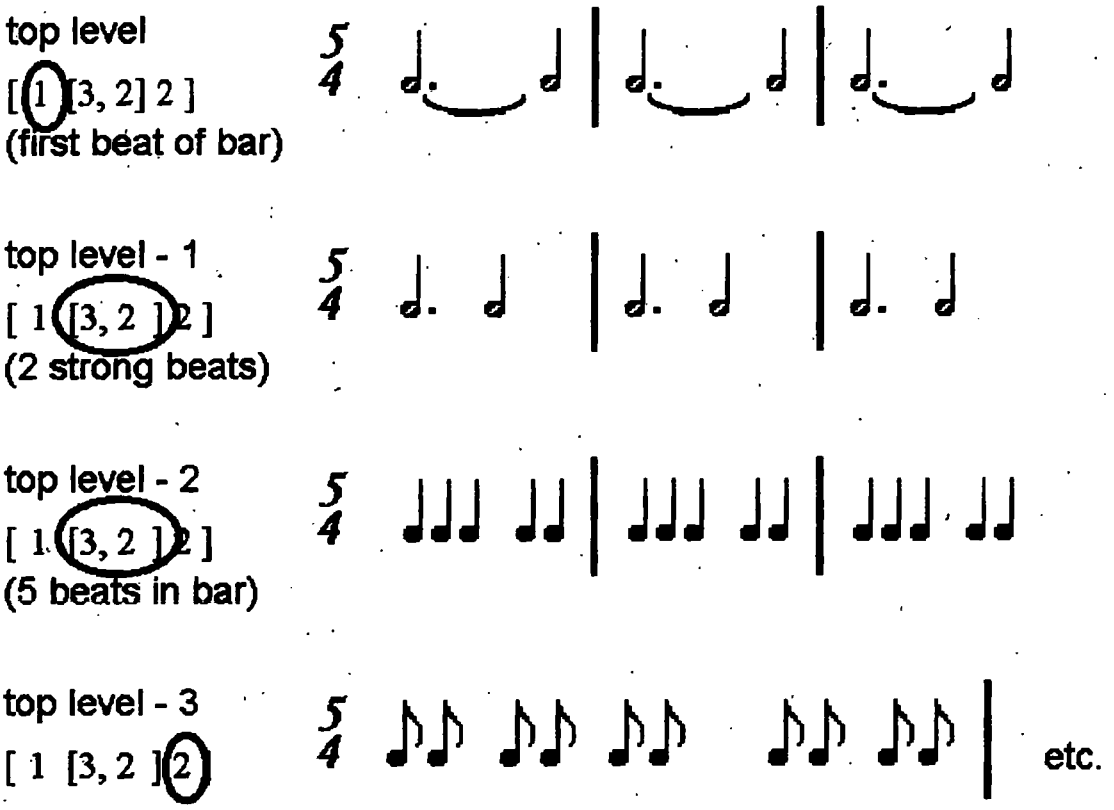

Figure 7.12: Hierarchical metric constraints for metric vector [1, [2, 3], 2].

Rationale behind metric constraint design

An implicit criterion (resulting from using *Implication-Realisation Model* analyses for generating analysis) is that for each hierarchical melody generation constraint the resulting analyses from the parser should bring to light the constraints in each level — thus each note that appears in a given level in a generating hierarchy (above the musical surface) should also appear in the corresponding level of a hierarchical analysis of any generated melody. The criteria led the design of the durational constraints leading from metric vectors.

As discussed in Chapter 3, for each group of notes making up a structure in an *Implication-Realisation Model* analysis, the initial and terminal notes are promoted. A question not explained by Narmour, nor represented in his graphical notation for analyses, is how the "gap" of the notes not promoted is filled. In Chapter 4 for the formalisation of the theory the action to be taken was that the promoted note has the duration of any non-promoted notes following it added to its duration (up to the next promoted note). This describes what happens when creating an analysis from the bottom up, with non-promoted notes being removed at each new level. As described earlier this chapter, when generating notes from the top-down, a note at one

level will have its duration reduced to allow the insertion of new notes at the levels below (for example the reduction of note "n1" from semibreve to minim to allow the insertion of note "n1-1" at level "top level -1", and the reduction from minim to crotchet to allow the insertion of note "n1-0-1" at the next level "top level - 2"). This demonstrates how the metric constraint-generation design was led by the formalised bottom-up analysis theory. The further work section of Chapter 8 discusses how both M-PARSER and M-CONSTRAINT could be changed if a different approach were taken, such as a note maintaining its musical surface level duration at higher levels in an *Implication-Realisation Model* analysis.

Harmonic constraints

There are two forms of harmonic constraints, which can be categorised into "vertical" (hierarchical) and "horizontal" constraints. Since the vertical harmonic constraints relate to the horizontal ones, the horizontal are described first.

Horizontal harmonic constraints

Chapter 5 and Appendix D describe in detail the representation of the harmonic aspects of a melody, as a list of the scales and chords that apply at any given point in the melody. In M-CONSTRAINT a melody is generated to a given metric and harmonic framework. The previous section has described how metric constraints can be applied hierarchically following from the hierarchical representation of metre as metric vectors. However, the representation of the harmonic framework of a melody is in no way hierarchical — it is simply a list of the scales and chords that apply to each note in the melody.

Scale and chord sequences can be generated hierarchically in a number of ways. Two possible methods are:

- **re-write grammar** — previous work (for example Ulrich 1977, and Steedman 1983) have demonstrated how the use of relatively simple re-write grammars can be used to generate chord sequences; such grammars can be used hierarchically by providing the starting and ending chords, and stating the number of intermediate chords to be generated, and
- **a hierarchical planner for paths in Holland's Harmony Space** — Holland (1989) describes a hierarchical planner for creating sequences of chords by describing paths in a two dimensional chord "space".

Both of the above methods allow the generation of a chord sequence in a top-down fashion, so that just as new notes may be chosen to be inserted in a melody, new chords associated with the new notes can be chosen. A third possibility, briefly outline in Chapter 8, is the use of a hierarchical planner based on creating scale/key and chord sequences from Lerdahl's pitch spaces. Currently no hierarchical methods for generation of scale/key sequences has been included in the design of M-CONSTRAINT, although either of the two techniques for chord sequences could be modified for such use.

Vertical harmonic constraints

The vertical harmonic constraints define the permitted pitches for a note at any given hierarchical level (in terms of consonance with the scale and chord associated with the "vertical slice" of the melody in which the note occurs). The vertical harmonic constraints used in M-CONSTRAINT are based on the "Pitch Space" description of Lerdahl's (1988). Figure 7.11 shows the five pitch spaces proposed by Lerdahl. The numbers in the lists are degrees of a scale. As one moves up the figure from Space 5 to Space 1, the degrees of the scale for each space are more consonant.

Space 1:	Tonic Space	[1]
Space 2:	Open Fifth Space	[1, 5]
Space 3:	Triadic Space	[1, 3, 5]
Space 4:	Diatonic Space	[1, 2, 3, 4, 5, 6, 7]
Space 5:	Chromatic Space	[any pitch]

Figure 7.13: The five pitch spaces proposed by Lerdahl (1988).

Lerdahl's pitch spaces can be used as the basis for vertical harmonic constraints for five hierarchical levels — the vertical harmonic constraints corresponding to the scale that applies to the vertical slice of the melody being generated. Figure 7.13 illustrates the set of Lerdahl's pitch spaces for the major chord which has the root the same as the scale (i.e. degrees [1, 3, 5] of a scale). M-CONSTRAINT is designed to allow different chords to apply to a given vertical slice of a melody, the general version of Lerdahl's pitch spaces used by M-CONSTRAINT, is presented in Figure 7.14.

Space 1:	Root Space	[cn1]
Space 2:	Root and Fifth Space	[cn1, cn5]
Space 3:	Chord Space	e.g. [cn1, cn3, cn5]
Space 4:	Scale Space	[sd1, sd2, ... sdN]
Space 5:	Chromatic Space	[any pitch]

Figure 7.14: The pitch spaces used by M-CONSTRAINT.

The list elements in Figure 7.14 notated "cn1", "cn3", and "cn5" refer to notes of the chord that applies for the vertical slice of the melody, for which vertical harmonic constraints are to be defined. Space 3 "Chord Space", contains those notes of the chord, which, if a triad, would be [cn1, cn3, cn5] as given, however the space also describes non-triadic chords (such as seventh chords, which would be notated [cn1, cn3, cn5, cn7]). The "Scale Space" is a list of degrees of the current scale (notated [sd1, sd2, ...]for scale degree one, scale degree 2 etc.).

7.3.5. Efficiency considerations for constrained melody generation

Techniques for improving the efficiency of constraint-satisfaction can be classified into two categories: those that attempt to resolve conflict between constraints after values have been assigned to variables (post-conflict techniques), and those that attempt to avoid as many constraint conflicts before values are assigned to variables (pre-conflict techniques). A number of forms of each class of technique are briefly discussed below[4].

Post-conflict techniques

Post conflict approaches are generally versions of the technique called dependency-directed backtracking (see, for example, Stallman and Sussman, 1977)[5], in which the choice of variable to change is directed by only choosing variables which are involved in the constraint conflicts — usually the choice is of a variable involved with the most recent constraint to be violated. If

---

[4] A summary of many of the constraint-satisfaction techniques we mention can be found in Guesgen and Hertzberg's publication (1991).

[5] Specific instances of dependency-related backtracking are *backjumping* (Gaschnig, 1979) and *graph-based backjumping* (Dechter, 1990).

all constraints relating to a variable are tested *immediately* when the variable is assigned a value, much assignment of other variables to inconsistent values can be avoided.

Pre-conflict approaches

Conflict avoidance is obviously an ideal approach to constraint-satisfaction — Guesgen and Hertzberg (1991) categorise pre-conflict approaches as either *look-ahead* or *heuristic based selection*. Look-ahead approaches include techniques such as "forward checking" (Haralick and Elliott, 1980) and filtering. Probably the most famous filtering algorithm is Waltz Filtering (Waltz, 1972), variants include Directional Arc Consistency (Dechter and Meiri, 1989) and structured tagging (Freuder, 1978).

Heuristic pre-conflict approaches include the following: *maximal reduction of search space* (i.e. choose the constraint or variable value that maximally reduces the search space); order constraints and variables[6] according to minimal (average) width (where the width is a measure of the number of connected constraints and variables earlier in the sequence, (Freuder, 1982)). Other heuristics are concerned with the *ordering of the domains for variables* (e.g. when a variable is to be assigned the domain can be ordered so that the choice is for values that offer maximum number of options for future variable assignments, see for example Dechter and Pearl, 1987), and *hill climbing* approaches where an initial set of variable assignments is made, and then variables are changed in order of maximum number of conflicts reduced by each change.

Heuristics for simple melodies

Since the current implementation of M-PARSER (and the prototypical procedures for M-CONSTRAINT) are written in Prolog, efficient code becomes a factor since real time response is a requirement of a direct manipulation tool such as MOTIVE. The constraints that make up a generating analysis are structured such that the end points of structures represent relationships between sets of constraints — i.e. each structure has a set of constraints about first and last notes, number of notes, contours and intervals (from the structure class) and any

---

[6] Guesgen and Hertzberg (1991) propose the concepts of *dynamic constraints*, where the association of domains for variables, and relationships between variables are both represented using a single formalism — thus choice of variable to assign a value to, and variable-relation constraint to test, become the same form of decision.

associated metric and harmonic constraints for the duration of the notes of the structure, the terminal notes of a structure also represent relations between different structures at the same level of an analysis, and between structures at different levels of an analysis. It seems likely that some advantage could be made of this by finding sets of values for the terminal notes of structures at all levels in an analysis before considering the additional notes within each structure.

7.4. Limitations of MOTIVE

The current programs forming the implemented prototype of MOTIVE have a number of limitations. In addition there are some limitations with the design of the MOTIVE tool itself. The following subsections discuss each of these two categories of limitation, and suggest ways to overcome them where appropriate. Some of the limitations discussed in this section are also addressed in the further research section of the final chapter of this thesis.

Limitations of the current prototype programs

Currently in the design of M-CONSTRAINT there is no account taken of ties between notes. The form of harmonic and metric constraints is simple compared to the sophistication of knowledge experienced composers and analysts apply to composition. However, by the fact of their simple nature, the constraints presented for M-CONSTRAINT mean that the understanding and application of them is an achievable task for novice composers.

The current prototype of MOTIVE is made up of a number of separate modules, for example modules for the generation and testing of metric, harmonic and *Implication-Realisation Model* structure constraints. At present not all of these modules have been fully integrated, so for example the Swan Lake generations used as examples earlier in this chapter were created by running a number of the modules separately, and then combining the results of these runs by hand to construct the actual examples presented. The two main restrictions to date in the integration of these modules have been:

- efficiency considerations — the amount of backtracking involved when all constraints are being evaluated using naïve generate-and-test procedures is excessive in terms of both time and memory (for the desktop machine used for the prototype implementation). With the application of one or more of the efficient constraint generation techniques reviewed earlier in this chapter these limitations

could be overcome.

- the development of the different MOTIVE modules have taken place over a period of years, and some problems of configuration management have been encountered. For example, the module for generating and testing metric constraints was the first to be developed, and assumes a simpler melody and analysis representation than has been finally developed. The process of bringing all modules up to date for integration is currently being undertaken.

Theoretical limitations of the MOTIVE tool

Beyond the minor implementational limitations of the prototype of MOTIVE are two deeper limitations inherent in the direction the design of the MOTIVE tool has taken. These two problems are discussed below. The research involved in overcoming these limitations is beyond that described in this thesis, although in addition to identifying the limitations here, suggestions are made for related further work in the next chapter.

The first of the problems with the present design of the MOTIVE tool is related to the number of possible solutions the tool may generate for a given generating analysis. As identified during the discussion of the alternative solutions to the Swan Lake generation earlier in this chapter, if a constraint network is underconstrained the number of possible solutions can be very large. For domains where any solution will do, this might be solved by simply choosing the first to be generated, or the first solution when solutions are ranked according to some metric of preference. In the case of the presentation of a "family" of generated melodies to a novice composer, however, having a large number of solutions is a problem, since the composer needs to know about the generated melodies in order to be informed for either the choice of one solution (according to the composers internal "metric") or to use knowledge of properties of the size and members of the solution set to go back to the constraint network for changes and regenerations from a modified generating analysis. In an ongoing research project (Smith 1995) this problem is addressed. In a similar manner to the way, for example, an efficiency metric might be used for the selection of a solution for a non-creative problem expressed as a constraint network, some more appropriate form of (aesthetic) metric might be used to assist a novice composer reduce, order or navigate through a large solution space. A metric that is constructed by the user, or perhaps has parameters that can be changed may have educational benefits — further work to investigate the form and use of metrics as a way to partially solve

the problem of large solution sets is outlined in the next chapter.

The second of the theoretical limitations of the MOTIVE tool is rooted in a problem with Narmour's presentation of the *Implication-Realisation Model*. The problem is the application of *Implication-Realisation Model* analyses to large melody fragments or to complete melodies themselves. We do not claim that the *Implication-Realisation Model,* or the MOTIVE tool, constitute a method for the analysis of large scale, complete melodies. Narmour's main claim is that, assuming the cognitive foundations of his theory are correct, his theory and the resultant analyses represent a model of the analytical structures built up by a listener upon hearing a melody[7]. For analyses based on Narmour's theory (and analyses and generations formed using the MOTIVE tool) to be applicable for pieces of music larger than the fragments used for illustration in this thesis and by Narmour some additional, larger scale framework would be required. Whether a successful larger scale framework would be from the top-down aspects Narmour has been more recently suggesting (outlined in Narmour 1992, and to be published in additional promised volumes), or from the application of other's music analysis theories is an area for further research.

What we do claim for the work in this thesis, is that the design of the MOTIVE system has been based upon goals of making the analysis and generation of simple melodies, and fragments of melodies, accessible to novices. MOTIVE aids students to undertake these tasks by using a theoretical framework that both concentrates on intrinsically melodic aspects of melody (rather than harmony as in the case of Schenkerian analysis, for example) and assists novice composers in the understanding of the relationships between melody and other musical concepts. Although the MOTIVE tool may not be appropriate for large scale melodies, it does present and allow the manipulation of melodies in terms of concepts such as contour and interval, and highlights the relationship of metre and harmony to melody through the definitions of closure due to these parameters.

---

[7] We shall not enter the debate as to whether it is appropriate to analyse a common music notation (i.e. an artificial symbolic) representation of a melody. Smoliar is one who has argued against this approach (e.g. in his review of the *Implication-Realisation Model*, 1991*)*.

7.5. Conclusions

In this chapter we have presented a detailed description of MOTIVE, the central component of a constraint-based intelligent learning environment for melody composition (an outline architecture of an ILE which would be built around MOTIVE is discussed in the next chapter). We have outlined in detail how MOTIVE has been prototypically implemented, using constraint-based extensions (M-CONSTRAINT) to our existing implementation of a parser for Narmour's theory (M-PARSER). Earlier this chapter we detailed the conclusions of the benefits of such an approach, which can be summarised as follows:

- students can work on either local or global aspects of the melody (by hierarchy, and use of vertical or horizontal constraints),
- students can use MOTIVE in both directions (i.e. a sequence of parse and generate) both on complete melodies, or fragments of melodies such as figures and phrases (this is possible because of the full implementation of the parsing process — had M-PARSER been implemented to analyse melodies in terms of complete structures only, this would not be possible), and
- a novel application of Lerdahl's (1988) Pitch Spaces for harmonic constraints for generation, has been presented.

We have presented a number of forms of constraints for the generation of melodies — some directly from the formalised *Implication-Realisation Model*, and others from existing research into metre and harmony. These constraints do not have a formal, music-theoretic justification, but seem sensible ways of allowing a student to reduce the generation-space. We have shown how the M-CONSTRAINT approach allows students to focus on particular aspects, of particular local parts of a melody (by looking at vertical constraints at given levels in the generating analysis)[8].

Our approach applies the AI technique of "generate and test" in two ways; first, the constraint-based tool takes an analysis (with additional constraints) and uses traditional generate and test

[8] In the further research section of the next chapter we propose how students can take a larger scale view of the melody as a whole (through the use of horizontal harmonic, and phrase-variation constraints, and by working on the constraints of the highest levels of the hierarchical analysis).

procedures to generate melodies to fit the analysis; second, a novice composer can use the constraint-based tool to generate a "family" of melodies that all fit a given analysis, and then choose one from those generated — thus there are two stages where melodies are tested, by the constraint-based tool itself, and by the user of the tool.

Small melodies and melody fragments have been generated by M-CONSTRAINT, i.e. they have been formed using a tool based on the *Implication-Realisation Model*. The feature of being able to move in both directions when constructing a generating analysis allows students in any educational system based on the Narmour's theory to continually associate concrete melodies (and properties of melodies) with the generating analyses on which they work. We have also demonstrated how a novel application of Lerdahl's *Pitch Space* can be used for implementing hierarchical, harmonic constraints.

In summary, we have designed and prototypically implemented a constraint-based generator that provides the central, essential building block around which an intelligent learning environment for novice composers of melody could be built.

Chapter 8:
Conclusions, critique and further work

This chapter presents a summary of the conclusions drawn in the preceding seven chapters; in addition some new conclusions about the contribution of the work are also described. A critique of the research is presented, and a number of detailed suggestions are made for further work leading from the research described in this thesis.

8.1. Contribution

8.1.1. Overview of contribution

Narmour's theory has provided musicologists with a sophisticated tool for the analysis of melodies. Although broad and generally described in much detail, the *Implication-Realisation Model* has a number of omissions, and a few inconsistencies (a summary of which we present in a moment). For the long term goal of the development of computer-based tools and learning environments for assisting novice composers we chose Narmour's theory, its strengths including a combination of its detail of description, its psychological basis, and its unique approach in a music theory by focusing on the perceptual structures listener's form when hearing a melody. However, the *Implication-Realisation Model* required formalisation before a computational model could be built. This thesis has described how we went about this formalisation process, using related work from researchers such as Lerdahl (1988) and Levitt (1985) to develop an unambiguous computational representation of the theory. This parser is (to our knowledge) the first large-scale, computational implementation of the *Implication-Realisation Model*.

It has required some work to formalise Narmour's theory, and an alternative methodology could have been to select a number of existing formalised theories (of music perception, of harmony and metre etc.) and attempt to combine them in some way as the basis for a constraint-based tool. However, there were two reasons why the choice of Narmour's theory was more appealing. First, there are no existing formalised or computational theories that model the analysis of music with an intrinsically melodic focus; even were one to be found it would be difficult to combine the different formalised aspects of music as they relate to melody into as consistent a model as the *Implication-Realisation Model*, which is able to focus on melody and still take account of other aspects of music in a single, consistent way. Second, although it might be possible to combine an existing set of theories in a musically plausible way, such a construction would then need to be presented and tested by the musicological community, in addition psychological testing would need to be performed to ensure the different theories were being combined in a reasonable way; Narmour's theory has already been extensively open to criticism, and while having its share of critics is generally accepted as a useful addition to the set of substantial music analysis theories available to musicologists.

Further to our long term goal, we have provided groundwork for the development of educational environments by extending the computational parser with constraint-based techniques, to allow the artefact of a melody analysis to form the basis of a constraint network for melody generation.

This thesis shows how a constraint-based AI approach (following that of Holland, 1991) can make use of an *analytical* theory of melody for the development of an aid for composition — a *generative* task. MOTIVE has been designed so as to make the task of analysis and generation of simple melodies a progressive task (i.e. by the gradual removal of default constraints and decisions, thus slowly more of the compositional task falls onto the student).

There is support for a constraint-based approach to creative tasks in general (e.g. Johnson-Laird, 1988), and that musical composition tasks involve the creation and application of constraints (e.g. Reitman 1965, Butler 1992). We have identified that the *Implication-Realisation Model* is a strong theory of melody. MOTIVE presents a unified approach to the manipulation of musical constraints in the general and intrinsically melody-theoretic based framework of the *Implication-Realisation Model*. Thus the work in this thesis provides the groundwork for future work to be undertaken to determine how successful intelligent learning

environments based on Narmour's theory can be (such as the ILE architecture proposed at the end of this chapter).

The contributions of the research described in this thesis are presented under the following two headings: contributions to AI & Music, and more tentative contributions to Music Education, based on the educational potential of a full version of our MOTIVE tool.

8.1.2. Contributions to AI & Music

Chapter 4 of this thesis presents a formalisation of Narmour's *Implication-Realisation Model*, and Chapters 5 and 6 respectively describe the computational model of the formalisation, and a testing of the formalised theory. The result of the research described in these chapters is the unambiguous representation of those aspects of the theory that form the computational model (M-PARSER), and a formal description of other aspects of the theory in the predicate calculus descriptions in Chapter 4. The research in this thesis makes two main contributions to the field of music theory: first, the formalisation of much of the bottom-up aspects of Narmour's *Implication-Realisation Model*; second, the development of a computational model of the theory, providing a tool for testing how changes in aspects of the theory affect the analyses of melodies (for example the examination of how changes in the respective importance of different forms of closure affect promotion and chaining in analyses).

Through the process of formalising the *Implication-Realisation Model*, and encoding the formalised version of the theory, we have uncovered and suggested solutions to a number of inconsistencies and omissions in Narmour's version of his theory. These contributions can be a summarised under the three headings: (i) aspects of Narmour's theory formalised, (ii) aspects of Narmour's theory encoded in M-PARSER, and (iii) omissions and inconsistencies identified. Each is summarised below.

(i) Aspects of Narmour's theory formalised

(1) The two rules of inference ("hypotheses" of continuation and reversal).

(2) The parametric scales for the melodic dimensions of interval, contour, and duration.

(3) The intervallic and contour constraints defining each class of structure.

(4) The following three forms of closure: stopping, metric, and durational.

(5) Rules for the joining (chaining) of structures when weak closure occurs.

(6) Rules for hierarchical promotion.

(iii) Aspects of Narmour's theory encoded in M-PARSER

(1) All of the above six features of Narmour's theory.

(2) The steps of the process of parsing when creating an analysis based on Narmour's theory — in the form of a set of control procedures and parser actions.

(iii) Omissions and inconsistencies identified

(1) Metre is not defined in any way in Narmour's theory. In M-PARSER metric strength is calculated from a strictly hierarchical representation of metre (based on Levitt (1985)),

(2) Harmony is not defined in Narmour's theory. We have modelled harmonic closure by applying Lerdahl's (1988) pitch space research in a novel way.

(3) Narmour's published analyses have been generated by a note-by-note process of parsing a given melody. This process is not described by Narmour, but has been formally modelled by the `"meta-parse"` procedures in M-PARSER (the control routines discussed in Chapter 4).

(4) The recursive application of Narmour's theory at successively higher hierarchical levels is not fully described by Narmour. A number of alternatives have been formally presented in Chapter 4 (for example replacing notes not promoted with rests of equal duration, and the freezing of metric strengths during hierarchical promotion), and one alternative has been implemented in M-PARSER.

(5) The rules for when notes are (in effect) promoted two hierarchical levels are not defined by Narmour, though examples of such promotions can be found throughout his publications (for example: Figure 10a and 10b, Narmour (1989); Ex. 21.4, 22.8a, 22.8b and 22.13, Narmour (1990)). Such cases are discussed in Chapter 3 of this thesis, and a more consistent way to parse, resulting in equivalent analyses has been defined.

Contributions from constraint-based extensions to the parser

One of the novel aspects of the work described in this thesis is the way a constraint-based

approach has allowed a tool for melody generation (i.e. synthesis) to be developed from a tool for melody analysis. There are two aspects to the constraint generation modules in MOTIVE, described below.

(1) The generation of melodies from a given Narmour analysis

One set of constraints used by MOTIVE is that notes must be generated to fit a specified sequence of structures from the *Implication-Realisation Model* — i.e. a melody can be generated by constraint-based techniques given the desired *analysis* one wishes the melody to adhere to.

This technique (given computational models expressed in ways amenable to contraint-satisfaction manipulation) could be applied equally well to other analytical theories of melody and music (for example the hierarchical analysis theories of Schenker, 1979, and Lerdahl and Jackendoff, 1983), and also to other domains where hierarchical analysis techniques exist (grammatical parallels with natural language suggest themselves, although the generation of interesting natural language from grammars may be less successful than of music, until more consideration of semantics can be encoded in the grammars, Walker *et. al*, 1987).

(2) The application of metric and harmonic constraints at different hierarchical levels, while generation of a melody from an analysis takes place

In addition to constraining the set of melodies to be generated by specifying the generated melody's analysis based on Narmour's theory, MOTIVE also demonstrates the application of harmonic and metric constraints at the different hierarchical levels of the analysis.

Although the current implementation of these constraints is a relatively simple use of Lerdahl's (1988) Pitch Spaces as harmonic constraints, and a strictly hierarchical view of metre similar to that of Levitt (1984), the use of similar constraints by themselves has been previously demonstrated for melody generation (Smith, 1990). The combination of the harmonic and metric constraints allows a reinforcement to the novice composer of the importance of higher levels in the hierarchical analysis — MOTIVE demonstrates how the AI technique of constraint-satisfaction can be used both to generate melodies from their analyses (as described above), and how the analyses themselves can be restricted to only include metrically and harmonically important notes in the higher levels of the hierarchical analysis.

In summary, the contribution of the research described in this thesis to the field of AI & music is the formalisation of, identification of ambiguities within, and computational representation of Narmour's *Implication-Realisation Model* of melody. In addition it has been shown how a melodies can be generated in terms of the *Implication-Realisation Model* and the application of additional harmonic and metric constraints.

8.1.3. Contributions to Music Education

The motivation of the research which this thesis describes is to provide the groundwork for the development of a computer-based educational system for novice composers of melody. We have prototyped MOTIVE, a tool for novice composers (eventually to be a part of a larger educational system), facilitating the performance of the following tasks:

(1) analysing an existing melody (in terms of grouping of notes according to Narmour's definitions of closure, and in terms of which notes are heard by a listener to be most important),

(2) generating a new melody (to meet specified constraints),

(3) completing a partially finished melody (with specified constraints), and

a full version of the MOTIVE tool would also facilitate a fourth task:

(4) building a complete melody around fragments.

Although not part of the main goal for the research described in this thesis, MOTIVE can also be used to help music theory students learn about Narmour's theory.

A fully implemented version of the MOTIVE tool (with an appropriate interface) would have a number of strengths over pen and paper composition tasks for novices, these are listed below:

(1) The theoretical basis of MOTIVE in Narmour's *Implication-Realisation Model* leads to a view of composition that has much emphasis on the listener. For example, the concept of the influence of intra-opus style provides a theoretical justification for how the recapitulation section of music in sonata form works — the exposition being the part of the melody where a listener learns intra-opus stylistic features of the piece, which are then recognised in the final section.

(2) MOTIVE provides students with a vocabulary for the description of fragments of melodies — i.e. Narmour's structures.

(3) The concept of closure in the theory provides a context for the discussion of how harmony, metre and style influence how melodies are heard (and how they can be used to structure melodies into groups of notes).

(4) The hierarchical nature of MOTIVE, and the application of constraints at hierarchical levels, provides a context for the discussion of larger scale structuring of melodies — the use of phrase and variation constraints (once implemented), for example, could lead to the concepts of musical form being introduced to a student.

(5) no instrumental skills are needed (the internal representation MOTIVE uses for melodies can easily be translated into MIDI or general MIDI format for audio playing via a MIDI instrument — a prototype audio interface using Hypercard already exists),

(6) the "Paul Simon" (Holland, 1989) approach to composition is greatly facilitated by MOTIVE, providing a method of analysing an existing melody, and then generating new melodies based on the analysis (where the student can experiment with the parts of the original's analysis and constraints to use for each new generation) — this approach to composition will be siginificantly enhanced when more types of constraint are implemented.

In summary, the contribution of the research described in this thesis to the field of music education is the prototype development of a multi-purpose tool for melody (MOTIVE), based on a formalisation of a physchologically grounded and general theory of music (derived from an intrinsically melodic view of music). Later this chapter we suggest ways that future research in AI music education could make use of our work, including proposals for the design of a direct manipulation interface for the MOTIVE tool, and a outline architecture for intelligent learning environments (MELODY-ED), within which MOTIVE could be used educationally by novice composers of melody.

8.2. Criticisms & limitations

8.2.1. Limitations of MOTIVE as a tool for musical novices

In the version of MOTIVE described in this thesis, students will need to understand rudimentary common music notation. When compared to similar musical microworlds (such as

Holland's (1989) Harmony Space), students using MOTIVE must initially learn a number of aspects of *Narmour's Implication-Realisation Model* (such as the classes of structure, and ideas of closure and hierarchy). However, informal empirical evaluation[1] suggests that musical novices can pick up these ideas rapidly when they are reinforced with the outputs of MOTIVE (and the resulting melodies played in an audible form via a Hypercard module).

Two main limitations of MOTIVE were discussed in Chapter 7: the number of possible generations; and whether or not MOTIVE, or indeed the *Implication-Realisation Model*, can scale up to large melodies. The first of these limitations, could be turned into an educational opportunity, through the exploration by the student of aesthetic metrics for reducing the solution space, or by decision making about the modification or addition of different constraints at different levels. The second limitation, that MOTIVE might not scale up to large melodies, is related to the lack of any representation of top-down, higher level constraints, which would relate to form, and horizontal, phrase-variation relationships between different parts of an analysis for a melody. In addition to the further research suggested to tackle this problem, it might be that MOTIVE without any such constraints could still be part of an effective intelligent learning environment, where the student would have to be consciously attempting to satisfy global constraints, as well as instructing MOTIVE about the local constraints to be enforced.

8.2.2. Limitations of the formalisation of Narmour's theory

As highlighted in Chapter 4, there are a number of omissions in our formalisation of Narmour's theory (style, level skipping, accent closure etc.), however all of these omissions can be attributed to the fact that Narmour, while applying these aspects of his theory for his published analyses, has not described these features in any detail, if at all.

8.2.3. Limited implementation of MOTIVE as a tool for novices

The current implementation of MOTIVE is a prototype. Although an audio interface module has been developed (a Hypercard stack that can play a melody from an intermediate file

[1] Another research project would be to test student's learning and understanding of the basic directional and magnitiude concepts of contour and interval size, to guide the design of any such direct manipulation interface.

generated by MOTIVE), a full version of MOTIVE would need an interface that is easy to use for novices, such as the one proposed as a further research in the next section.

8.3. Further work

In this section we present eight projects for further research, leading on from the research described in this thesis. The projects are presented in approximate order of size: four small projects, followed by three medium size projects (of one or more years duration), and one large scale project:

§8.3.1. summary of extensions to the current implementations,

§8.3.2. implementation of additional forms of constraint,

§8.3.3. proposals for a direct manipulation tool for harmonic analysis and generation, based on Lerdahl's pitch spaces,

§8.3.4. a proposal for a direct manipulation interface for the MOTIVE tool,

§8.3.5. the analysis of a corpus of musical examples,

§8.3.6. the collaborative inference of musical constraints between learner and computational constraint inference system,

§8.3.7. research into the computational representation and learning of style,

§8.3.8. proposals for an architecture for MELODY-ED, a constraint-based intelligent learning environment for melody composition and analysis, making use of the MOTIVE tool.

8.3.1. Extensions to the current implementations

The metric and harmonic closure modules of MOTIVE could be made more sophisticated, by the implementation of procedures for more human-like harmonic and metric analysis. The harmonic analysis could be made more sophisticated by the extension of the current Lerdahl pitch space model to include the modelling of chord proximity across regions. An extension to the metric analysis component of M-PARSER could include a metric inference module — some preliminary prototypes have been begun, based on the metric analysis work of Longuet-Higgins and Lee (1984). An additional extension would be the formalisation and computational modelling of the octave, which to date has been simplified in our work to be considered as a unison interval.

8.3.2. Addtional forms of constraints

In addition to the vertical use of constraints discussed in the previous chapter, an important issue in composition (as identified in the more successful students from the studies of novice and expert composers reviewed in Chapter 2) is the need for students to keep in mind the overall structure of a piece, and to be able to satisfy multiple constraints, both locally and globally. Here we briefly examine a form of phrase-variation class of horizonal constraints that could be developed for use with MOTIVE (perhaps as part of an intra-opus style learning module), and how the combination of horizontal and vertical constraints would facilitate the switching between local and global views, which we describe as ***level navigation***.

Phrase-variation constraints (horizontal)

The constraints described above tend to focus on the *vertical* constraints for a particular, local part of a melody (i.e. attributes of notes at a particular point in time). Another group of important constraints relate to the overall structure of a melody — this is partly controlled through the hierarchical approach to melody generation, in that students have control of the shape of a melody's overall shape by controlling the structures used at the highest levels of the generating analysis. In addition, a set of phrase-variation constraints can be used to link the note values at the musical surface at different points in a melody, by constraining the notes generated for a particular structure (or sequence of structures) to have to conform to stated figures. The phrase relationship constraints could be made of figures of intervallic relationships, durational relationships, and specific pitch relations, making different parts of the melody use pitch transpositions, "double" or "half" speed, and exact repetitions of figures and phrases respectively. The phrase-variation constraints could be presented to the student in the form of a genealogy, i.e. the ancestry of a phrase or figure given in the form of a hierarchy, showing how it was originally, and the set of figures and phrases related to the original in branches showing how they change over time, and what form the changes take (such phrase genealogies could be based upon previous work in this area, for example Holland's (1991) PlanC system, and Smith 1990). Some form of overlay interface could be used to indicate how the phrase-variation constraints relate to the generating analysis under construction.

Level navigation

An important feature of the M-CONSTRAINT approach (derived from Holland's 1989 "MC" proposals) that makes such phrase-variation not only possible, but part of a consistent

formalism for applying constraints for melody generation is derived from the way one can work in both directions in the hierarchy. This use of the approach in both directions does not have to be so global, for example during the (downwards) construction of a generating analysis, a student may wish to introduce a particular phrase in the musical surface. This can be done by either stating an override constraint that at a particular point in the melody the phrase is to appear (perhaps forcing a relaxation of the analysis above that point), or (more consistently with the generating analysis approach), the phrase could be parsed (upwards), and then the resulting sub-analysis inserted into the lowest levels of the generating analysis under construction, with promoted notes from that sub-analysis being "bubbled up" the analysis as appropriate. Variations of the phrase would take the form of introducing altered versions of the sub-analysis at particular points in the generating analysis (so the use of an intervallic figure would be a relaxed sub-analysis, ignoring contour constraints but retaining pitch ones).

Towards a solution for the problem of too many generations

There are currently prototypical implementations of a number of the constraints discussed in this thesis. None of the efficiency techniques or phrase-variation constraints have yet been modelled. These prototypes have highlighted a problem of a large number of generations fitting particular sub-analyses, however the introduction of more constraints (perhaps arbitrary ones) could be used to reduce the outputs. Alternatively, by introducing a measure (or different measures) of similarity of melodies, the student could specify a maximum number of generations to be displayed, using the similarity measure to provide a set of generations that are most varied, but which still meet the specified criteria. The metric could be based on principles of aesthetics — the possible use of a metric for navigation and control of solution spaces in creative domains has begun to be explored (see Smith 1995 for example).

8.3.3. Direct manipulation harmony tool based on Lerdahl's pitch spaces

Lerdahl's pitch spaces form both part of the formalised harmonic analysis component of the parser, and are the basis for constraints used for the generation of melodies. A tool for harmonic analysis and generation, to be used in conjunction with MOTIVE could be based upon Lerdahl's theory. One aspect of the non-symmetrical, but repeating spaces, could be their presentation in the form of a direction manipulation tool, where the pitch spaces were arranged in the form of concentric circles, the innermost never moving (the chromatic space), and each outer space moving more than the previous. Thus a change of key would be seen by the user graphically as a rotation of all but the inner space, and a change of chord a rotation of the

triadic space ring. The ideas of skip and step, and movement between the spaces would appear to be intuitive. The design, prototyping, implementation and empirical testing of such a tool is a project currently beginning.

8.3.4. A direct manipulation interface for MOTIVE

It would be unreasonable to expect a constraint-based tool to be useful for students if the constraints had to be manipulated in technical, textual or numeric form for musical tasks. Therefore an interface with direct manipulation features is required. The product of the MELODY-ED system is a melody (or "family" of related melodies) — if the tool is to conform to standard music notation for melodies a graphical interface is required. The provision of an audio presentation of a melody is also important, because it is highly likely that the novice composers using the system are poor sight-readers, and not able to "hear" music reliably when reading scores (although the audio output of the system may be via a different program, and for example, an intermediate MIDI file). Both the benefits and requirements for direct manipulation interfaces have been researched (Shneiderman, 1982).

Here we summarise the requirements of the interface for MELODY-ED (and more specifically the MOTIVE tool):

(1) during construction of a generating analysis the analysis should be presented in a form that explicitly shows: the notes already generated, the class of each structure and notes each structure relates to, the relationships between notes at different levels, and the form, scope and position of any additional constraints,

(2) ideally, as soon as a constraint has been added, removed or changed the system will (in real time) regenerate one of the possible melodies, or state which constraints are involved in any conflict if a melody cannot be generated,

(3) the notation used for the melody should clearly show the onset time, duration and pitch of each note (i.e. common music notation is a promising candidate), and

(4) the notation used for the analysis should clearly show the attributes of the class of each *Implication-Realisation Model* structure (so Narmour's mnemonic one or two letter symbols are insufficient).

For this medium sized proposal a detailed framework is briefly outlined below. We now present a design for such a direct manipulation interface, and a number of examples of possible interface elements for a tool such as MOTIVE.

Narmour's one or two letter mnemonics for structures (P, VR, (IP) etc.), while a useful aid for music analysts using his theory, are not a particularly intuitive aid for novices using MOTIVE. For the tool to be easily used and understood, a more graphical representation could be developed, whereby the symbol (icon) referring to each structure contained elements representing the distinguishing characteristics of the structure. For example in the following figure (Figure 8.1) icons show how interval size and contour can be represented in a single icon (a double line means a large interval).

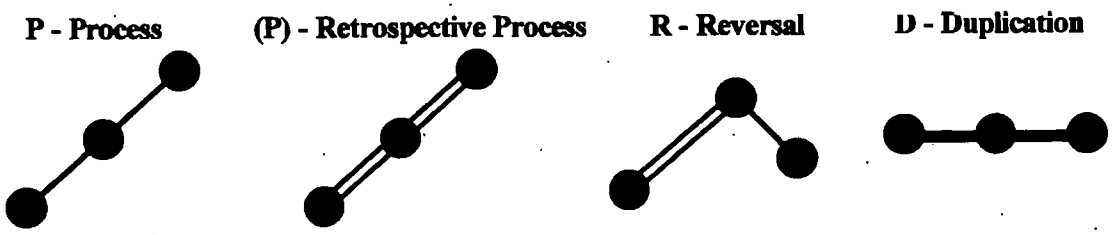

Figure 8.1: Examples of iconic representation of melodic structure classes.

Such ideas, along with more standard toolbox and menu-based approaches could be combined to form such an interface as that illustrated in Figure 8.2.

Figure 8.2 shows an interface with a number of components. The top of the figure shows icons for the nine structure classes, and one for creating retrospective structures. In the example the icon for a process structure - P in Narmour's notation - is highlighted, referring to the process structure highlighted in the third level of the hierarchical generation in the lower part of Figure 8.2. By choosing a different structure the student could get MOTIVE to re-generate the melody (also the number of notes in the structure is a choice the student would have to make, unless it is pre-determined by the constraints). The number of notes associated with each structure is explicitly represented by the notes that appear within the bounds of a structure icon. Two other icons allow a student to view (and change) the metric and harmonic constraints.

The interface in Figure 8.2 is illustrative of the kinds of interface components that could be used to provide interaction with MOTIVE, with a number of direct manipulation features. The example interface leaves a number of tasks incomplete. Such as how are phrase-variation

constraints represented in a direct manipulation form (they offer a more complicated problem because of the need to represent relationships between physically separate parts of a melody), such tasks we leave for future research.

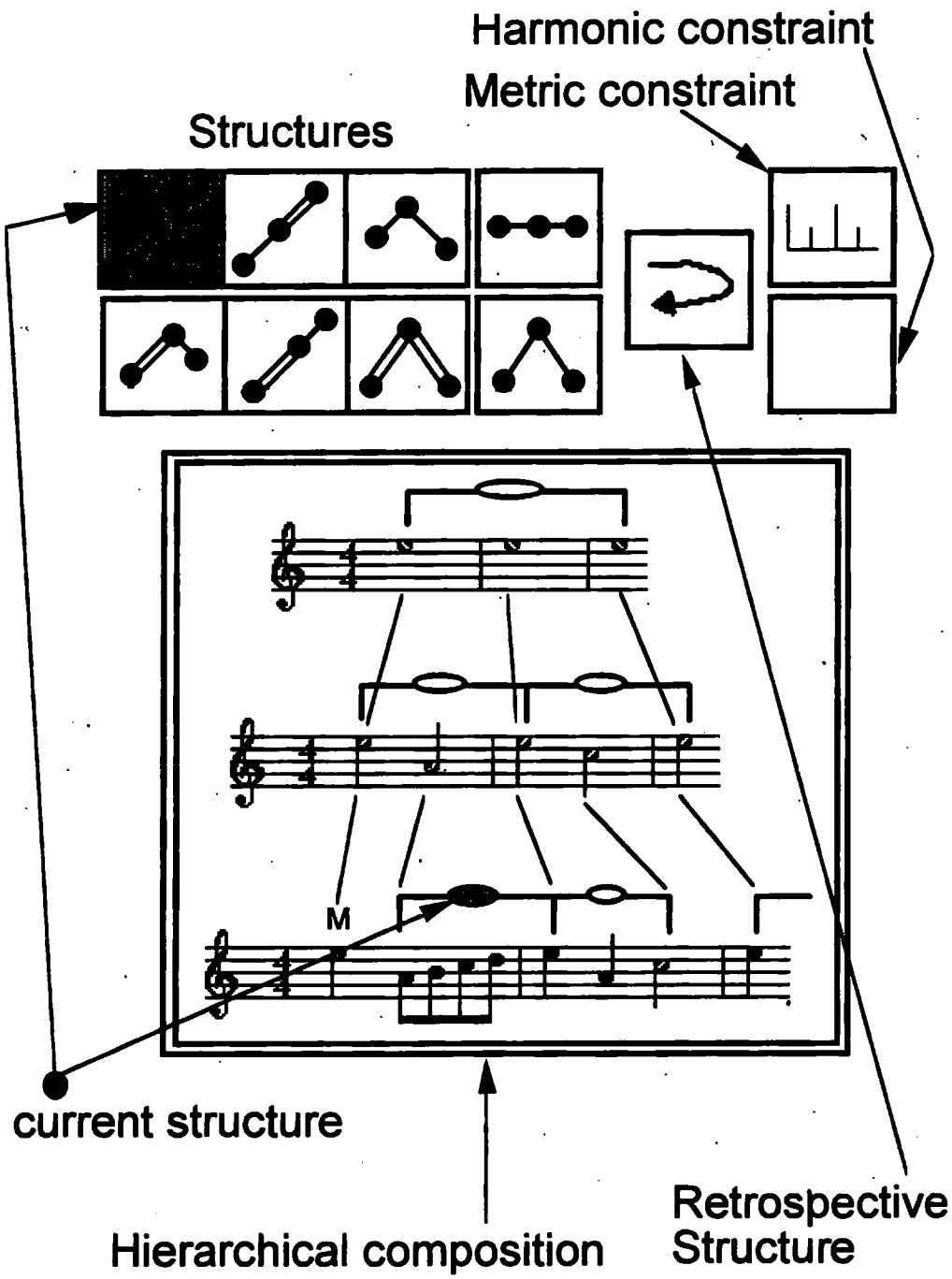

Figure 8.2: Key features of a possible interface for MOTIVE.

It is intended that MOTIVE be used in conjunction with tools for rhythm and harmony (for example Holland's (1989) *Harmony Space*), so that reasonably sophisticated constraints can be

chosen for MOTIVE to generate within. For example, a harmony tool could generate a chord sequence (perhaps including a modulation). This "horizontal" sequence of chords would become constraints for the "vertical" levels above each chord - i.e. if bars 3 - 5 had the chord Cmajor, then the Lerdahl pitch spaces for the hierarchical levels of bars 3 - 5 would refer to Cmajor.

A design for an iconic interface has been proposed, making strong use of the visual modality by designing graphical icons that, we suggest, provide an intuitive representation for the primitive melodic structures of Narmour's *Implication-Realisation Model*. Thus we have demonstrated how the development of an interface for MOTIVE, with many direct manipulation elements is straightforward in principle to implement.

An audio-graphical interface for MOTIVE is proposed in the next chapter, illustrating how many aspects of the *Implication-Realisation Model* embodied by MOTIVE can be clearly represented in a graphic modality, allowing students to use existing cognitive skills in the modality such as differentiating between the magnitudes and direction of objects.

8.3.5. Corpus analysis

A large corpus of melodies could be analysed to the frequency and hierarchical level of occurrence of each of the classes of melodic structure. From such analysis higher level constraints may come to light, which could then be compiled into heuristics to guide student's use of MOTIVE for top-down composition. Narmour begins to suggest such an investigation as follows:

> *"Since the bottom-up part of the theory is context-free, and applicable to all learned styles of melody, the model should enable us to discover what kinds of aesthetic strategies composers employ with respect to realisation, denial and thus, by inference, degree of idiostructural surprise. And from the economical symbology we should be able to learn (on all levels) what kinds of melodic strings occur most commonly."* (Narmour 1989, p. 57)

The similarity of analyses such as Figures 8.3 and 8.4[2] below, suggest that such corpus investigation as described above may be a useful technique for the classification of compositional styles and strategies.

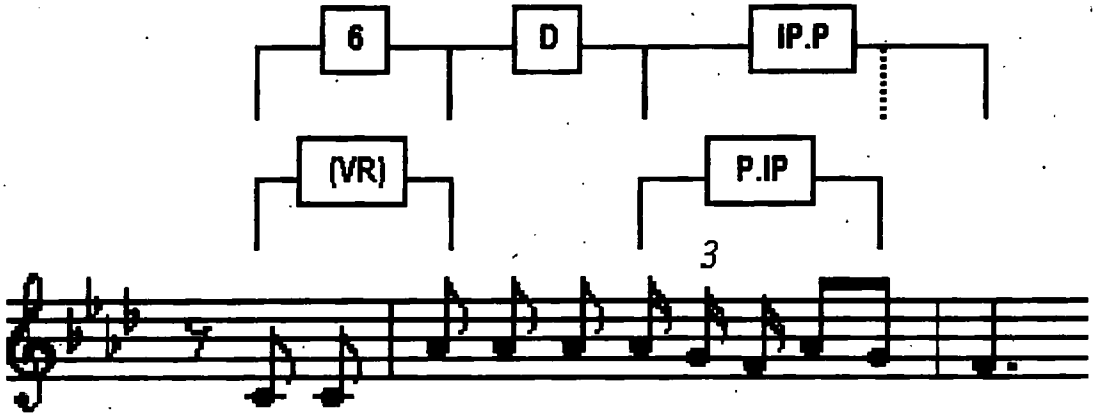

Figure 8.3: Analysis of Verdi, Un Ballo in Maschera, Act II, sc. 1.

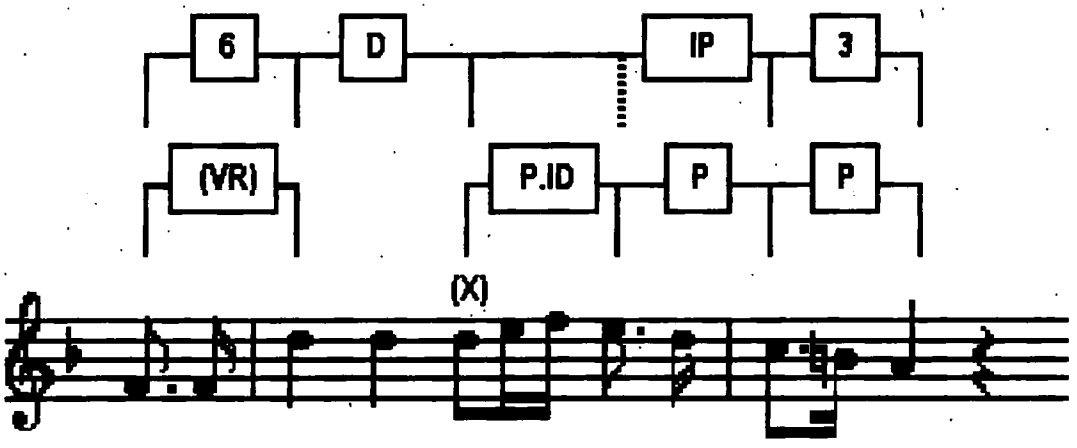

Figure 8.4: Analysis of Verdi, I Lombardi, Act IV, sc. 2.

8.3.6. Collaborative inference of constraints

A task which student, tutor and system could collaboratively perform in the MELODY-ED system we propose at the end of this chapter is the inference of new constraints to be added to a knowledge base of musical constraints. Students may wish to add new constraints to guide a composition which at that time would not be stored in MELODY-ED's constraint knowledge

---

[2] These figures appear as parts 'a' and 'b' of Figure 10, Narmour (1989).

base. The student would be given the choice of either simply adding new constraints to the knowledge base, or (preferably) entering into a negotiation process with the tutor and system, requiring the student to both define what type of constraint is to be added, and also to define where the constraint can be inferred from (e.g. by analysis of one or more existing pieces of music). The dialogue may lead to the choice of a relaxed form of the constraint if the tutor is unable to be convinced by the student.

8.3.7. Research into musical style

A major, but rewarding project would be research into both how to represent musical style, and to model how listeners both recognise styles they know and acquire knowledge of new ones. The two forms of style that form part of the top-down aspects of Narmour's theory, would each form a project in their own right. An "off-line" analysis of a corpus of pieces could be used as an approach for a listener's learning of extra-opus style. A real-time system would probably be needed to give users the benefit of an intra-opus style learning module, when the melody in question is constantly being changed.

8.3.8. MELODY-ED: A proposed intelligent learning environment

MELODY-ED is a proposed intelligent learning environment, based around the MOTIVE tool, for novice composers of melody. The constraint-based MELODY-ED architecture is based upon the MC framework (Holland 1989, 1990 and 1991). Novel aspects of our proposed architecture are use of automatic parsing, based on some of the same constraints a student may use for melody generation, and the collaborative inference of new constraints.

We have not aimed to present a prescriptive, complete system design, which guarantees certain educational benefits, but we do present all of the components which are the pre-requisites for an intelligent learning environment for melody composition based on the *Implication-Realisation Model*.

For this long term proposal a detailed framework is briefly outlined below. Before outlining the components and workings of MELODY-ED, a brief overview of the different tasks that could be performed using the system is presented.

Overview of tasks students perform in MELODY-ED

The two figures below (Figure 8.5 and Figure 8.6) graphically represent the two main tasks

that could be performed using MELODY-ED: (1) the parsing of existing melodies; and (2) the construction of a "generating analysis", to be used in the constrained generation of a family of melodies. Each of the figures is described in detail below. In the figures: ovals contain objects represented by the computer (e.g. a melody or an analysis of a melody), rectangles represent processes performed by MELODY-ED, and non-boxed text represent actions that can be performed by the student and tutor.

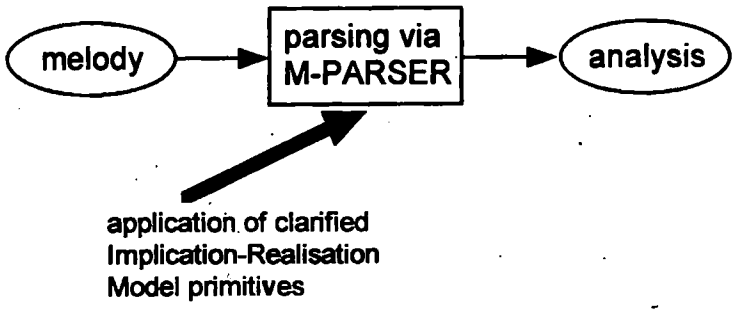

Figure 8.5: Illustration of parsing of a melody in MELODY-ED.

Figure 8.5 above illustrates the basic input and output of an analysis of a melody, the analysis being executed by M-PARSER (the details of this process are described in Chapter 5).

Figure 8.6 below is based on the concept of a "generating analysis" (i.e. the kinds of analyses described in Chapter 7, which can be represented as a constraint network for generation) — a "generating analysis" is an analysis in the form of one for the *Implication-Realisation Model*, which may have been constructed by the student, or be output from M-PARSER as the result of the analysis of an existing melody (or perhaps a student-modified M-PARSER analysis). There may also be additional constraints associated with the analysis. The generating analysis is given to a constraint-satisfaction module which outputs a set (or "family") of melodies which all conform to the analysis (i.e. if analysed by M-PARSER result in the analysis part of the generating analysis). This process views an analysis as a set of constraints on note "variables", describing only those melodies the analysis fits.

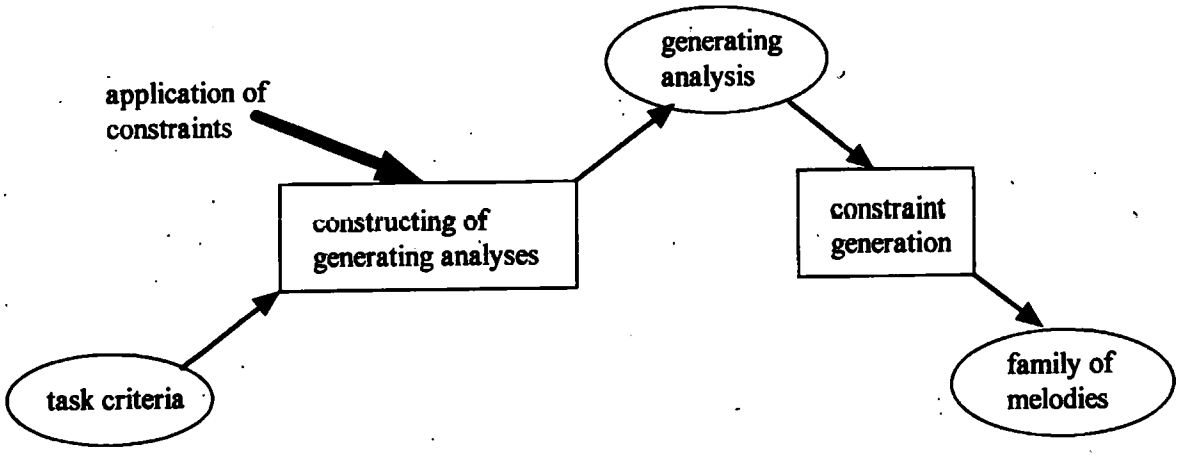

Figure 8.6: Illustration of construction of generating analysis, and melody generation using MELODY-ED.

As illustrated in Figure 8.6, some initial task criteria may be input (by either system or student, such as the time signature or number of bars the final melody is to fit with), and any present become part of the set of additional constraints on the generating analysis. The diagram shows that the construction of the generating analysis may be performed collaboratively[3] with the tutor.

MELODY-ED architecture attributes from Johnson-Laird's multi-stage model of creativity

The following is a description of each of the components of MELODY-ED pictured in Figure 8.7. The figure describes the architecture in terms of two agents: a student composer, and the tutor. In addition to the description of each component, a summary of the agent-based actions using each component is given.

Constraint sub-system

The constraint sub-system consists of three components: a *toolbox* of analysis and generation tools such as MOTIVE, a *knowledge-base* of musical constraints, and a *constraint engine*. The toolbox is envisaged to be a set of tools for music analysis and generation, tools such as MOTIVE, and Holland's *Harmony Space*. Each will be implemented in a constraint-based form, allowing an existing piece of music to be described in terms of a network of constraints

[3] That is the final choice of particular constraints would be decided by discussion between the tutor and student, either of which would be free to suggest or challenge the other's choices.

from a number of different musical *views* — each tool imposing a view of the piece according to the music-theoretical basis upon which the tool is based.

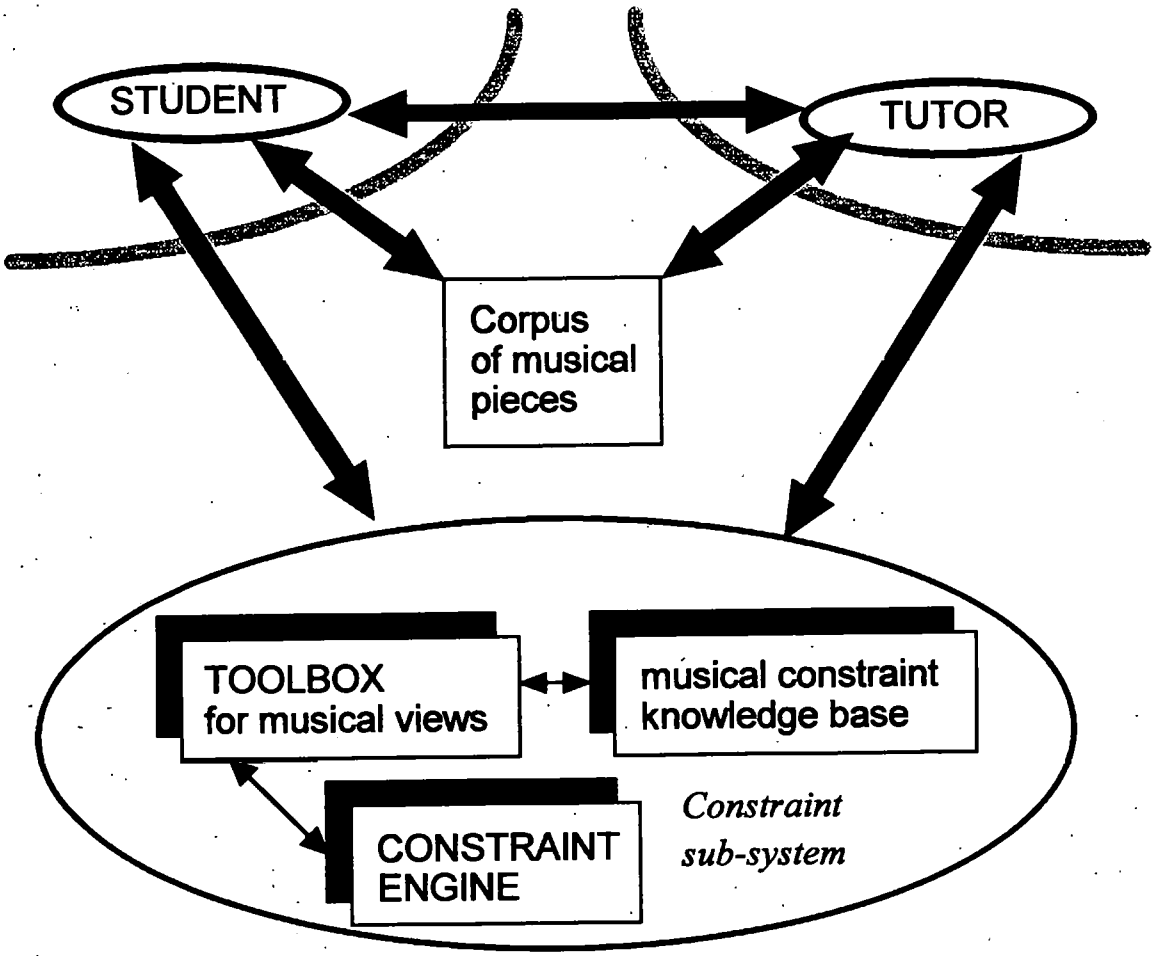

Figure 8.7: Architecture of MELODY-ED interactive learning environment.

The potential advantages of multiple-viewpoints in intelligent learning environments have been reported in existing research (for example, Cerri *et al.* 1986, Holland 1989), and would seem particularly important for domains such as music analysis due to the uncertain knowledge and multiple possible solutions identified by Baker (1992). In this case we are suggesting that different sets of domain knowledge can be used to present a variety of views of an artifact to a student, rather than using a set of different views of a single piece of domain knowledge as is the traditional use of multiple viewpoints.

Each tool in the toolbox uses constraints defined in the *knowledge-base* of musical constraints. These constraints are open to negotiation as to their form, when they can be applied, when relaxed or completely removed. New constraints can be added by the student, or co-operatively by student and AI tutor when inferring constraints from a piece (or collection of pieces) of music.

The third component of MELODY-ED's constraint sub-system is a *constraint engine*. This will take a set of variables, domains, and constraints describing relationships between the variables and generate interpretations (sets of values for the variables for which the constraints are satisfied). The tools from the toolbox use the constraint engine when analysing and generating melodies. This component is based upon a view of constraint generation as generally described by Levitt (1985), and described in detail as part of Holland's (1989) MC framework.

Tutor's tasks

The tutor has the main tasks of setting initial criteria for composition tasks, being available for discussion and negotiation if the student wishes to engage in such dialogue, and looking for actions by the student which the tutor's teaching strategy considers reason for initiating discussion and possible negotiation (for example when a student introduces a constraint for generation which conflicts with one of the overall composition constraints). Currently it is intended for the tutor to engage with the student during three activities:

- collaborative analysis of pieces of music,
- collaborative generation of pieces of music, and
- collaborative inference of constraints from existing pieces of music.

Corpus of Musical pieces

The analysis of a melody, and the inference of new constraints from existing melodies are both tasks that require a store of existing musical pieces. This module of MELODY-ED simply represents that store. At any time the student or tutor may add, remove, or change melodies in the store.

See Appendix H for a discussion of the feasibility of each of the components of MELODY-ED listed above.

8.4. Concluding remarks

The formalisations and computational models we have presented in this thesis provide groundwork for the development of intelligent learning environments for novice composers of melody. We hope that the work presented in this thesis will contribute in some way to the future development of Narmour's theory or other theories of melody. We also hope that eventually the ideas we have presented here could have an impact on music education.

List of appendices

Appendix A: Glossary of terms and symbols

(O) indicates weak resolution

O indicates moderate resolution

O indicates strong resolution

(×) indicates weak dissonance

× indicates moderate dissonance

⊗ indicates strong dissonance

aba exact registral return — a discontiguous sequence of three notes, such that the first and third notes are the same pitch, the middle note being different

aba[1] near registral return — a discontiguous sequence of three notes, such that the first and third notes are pitches within a major second of each other, the middle note being different

(b) indicates closure due to the influence of metre (the 'b' being a mnemonic for *beat*)

bottom-up the generation of an artefact (e.g. an analysis of a melody) working upwards from the bottom level of the hierarchy (e.g. starting with the melody itself), where the elements at a given level in the hierarchy are an abstraction derived from the level immediately below

closure the interactions of musical parameters causing inhibiting or weaking of melodic implication (causing the perceptual groupings of notes in a melody)

(d) indicates closure due to the influence of duration

D a duplication structure — a sequence of three or more contiguous notes with the same pitch

(D) a retrospective duplication structure — a sequence of three or more contiguous notes with the same pitch (only recognised as a duplication at level in the analysis higher than level 1)

(h) indicates closure due to the top-down influence of harmony

hypothesis there are two rules of inference to determine whether an interval will imply coninuation or reversal of the dimensions of melody — Narmour calls these two rules the **hypothesis of continuation** and the **hypothesis of reversal**

hypothesis of continuation this states that a small interval (see glossary entry for **small interval**) will initiate implications of continuation in a listener — i.e. implications that the contour for the next interval will be the same, and that the next interval will be similar in size (see glossary entry for **intervallic similarity**)

hypothesis of reversal this states that a large interval (see glossary entry for **large interval**) will initiate implications of reversal in a listener — i.e. implications that the contour for the next interval will be different, and that the next interval will be different in size (see glossary entry for **intervallic differentiation**)

ID a prospective intervallic duplication structure — a contiguous sequence of three or more notes, such that all the intervals are the same and small (though greater than unison), but the contour changes in an up/down or down/up sequence

(ID) a retrospective intervallic duplication structure — a contiguous sequence of three or more notes, such that all the intervals are the same and larger, but the contour changes in an up/down or down/up sequence

IP a prospective intervallic process structure — a contiguous sequence of three or more notes, such that all the intervals are the small (the first being greater than unison), but the contour changes in an up/down or down/up sequence

(IP) a retrospective intervallic process structure — a contiguous sequence of three or more notes, such that all the intervals are the large, but the contour changes in an up/down or down/up sequence

IR a prospective intervallic reversal structure — a contiguous sequence of three notes, such that all the first interval is large, and the second interval is small with the same contour

(IR) a retrospective intervallic reversal structure — a contiguous sequence of three notes, such that all the first interval is small, and the second interval is smaller still with the same contour

intervallic similarity when two intervals are defined as similar in magnitude according to a parametric scale

intervallic differentiation when two intervals are defined as different in magnitude according to a parametric scale

large interval an interval determined as large according to a parametric scale (important since large intervals trigger implications of reversal)

mL motion left on a parametric scale — i.e. motion towards closure. For the melodic dimension of interval this is an interval followed by a second, smaller interval. For the melodic dimension of contour, two intervals changing contour (i.e. one of the following: lateral/up, lateral/down, up/down, up/lateral, down/up, down/lateral)
NOTE: motion towards closure shows weakening of implication

mR motion right on a parametric scale — i.e. motion towards non-closure. For the melodic dimension of interval this is an interval followed by a second, larger interval. For the melodic dimension of contour, two intervals both ascending or descending
NOTE: motion towards non-closure shows strengthening of implication

mN non-motion on a parametric scale (i.e. for the melodic dimension of interval, two intervals of the same size; for the melodic dimension of contour, two unison intervals (so contour is lateral/lateral)

NOTE: non-motion shows no change in strength of implication

os indicates closure due to the top-down influence of intra-opus style

P a prospective process structure — a contiguous sequence of three or more notes, such that all the intervals are small (the first being greater than unison), and all contours in the sequence are the same (either all ascending or all descending)

(P) a retrospective process structure — a contiguous sequence of three or more notes, such that all the intervals are large, and all contours in the sequence are the same (either all ascending or all descending)

parametric scale an ordered sequence of values for a dimension of melody. For each scale movements to the left indicate weakening implication, and to the right strengthening implication. The two main scales used are:

(I) interval (where the scale has unison leftmost, and gets larger by a semitone each step to the right), and

(V) contour, a scale of pairs of contours — the scale has three divisions, the left containing

[lateral/lateral], the middle containing [ascent/ascent, descent/descent], and the right division containing the remaining possible pairs of contour [ascent/descent, descent/ascent, ascent/lateral, descent/lateral, lateral/ascent, lateral/descent]

R a prospective reversal structure — a contiguous sequence of three notes, such that all the first interval is large, and the second interval is small with a different contour

(R) a retrospective reversal structure — a contiguous sequence of three notes, such that all the first interval is small, and the second interval is smaller still with a different contour

small interval an interval determined as small according to a parametric scale (important since small intervals trigger implications of continuation)

structure the grouping of a sequence on contiguous notes (of a given level in an analysis), in terms of the implication and realisation of the melodic dimensions of interval and contour — the possible structures are: D, ID, P, IP, VP, R, IR, VR, (ID), (P), (IP), (VP), (R), (IR), (VR)

top-down the generation of an artefact (e.g. an analysis or melody) working downwards from the top level of the hierarchy (i.e. starting with the most abstract representation of the artefact), where the elements at a given level in the hierarchy are a refinement based on the level immediately above

VP a prospective registral process structure — a contiguous sequence of three notes, such that the first interval is small (although greater than unison), and the second interval is large, the contour of both intervals being the same (either both ascent or both descent)

(VP) a retrospective registral process structure — a contiguous sequence of three notes, such that the first interval is large, and the second interval larger still, the contour of both intervals being the same (either both ascent or both descent)

VR a prospective registral reversal structure — a contiguous sequence of three notes, such that all the first interval is large, but the second is small with a different contour

(VR) a retrospective registral reversal structure — a contiguous sequence of three notes, such that all the first interval is small, but the second is large with a different contour

xs indicates closure due to the top-down influence of extra-opus style

Appendix B: Introduction to melody

Although some music involves the interplay of many separable voices, Narmour's theory (as formalised in this thesis) considers only music with one melody playing. A simple definition of melody (ignoring harmonic and metric context) could be:

"A succession of single notes, varying in pitch"

Melody can be thought of as existing in two dimensions, pitch and time, as illustrated in Figure B.1.

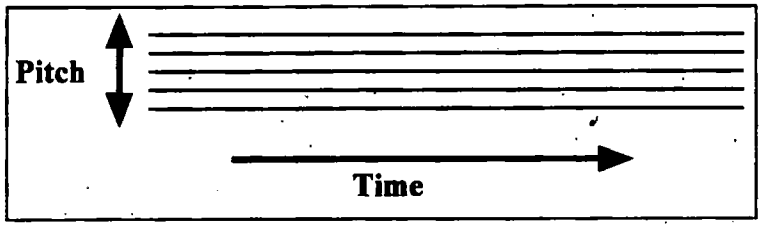

Figure B.1 ; Pitch and Time, the two dimensions of melody.

There are many parameters one can consider a part of melody, for example the timbre of the instrument and volume of the notes, however the Implication-Realisation Model mostly considers only three parameters,

pitch (and from it registral direction[1], defined below), onset-time and duration. Each of these parameters is defined in the theory, as are the parametric scales Narmour proposes with which to measure the parameters.

Pitch

Pitch describes the frequency of a note - how high or low it sounds. The melody of a piece of music usually falls within a span of two or three octaves, the range being defined by the singer or instrument sounding the tune.

In the Well-Tempered Scale of Western Tonal Music (the music my research and Narmour's theory consider), there are twelve different classes of pitch. Figure B.2 lists these pitch classes.

1	2	3	4	5	6	7	8	9	10	11	12
C	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
	D♭		**E♭**			**G♭**		**A♭**		**B♭**	

Figure B.2: The twelve pitch classes of Western Tonal Music.

As can be seen, five of these pitch classes have two names (numbered 2, 4, 7, 9 and 11 above) - one with a sharp sign (#) and the other a flat sign (♭). These two signs are known as accidentals. In the Well-Tempered Scale both names (for example D# and E♭) represent the same physical frequency, but indicate different relationships to other notes.

These twelve pitch classes are repeated (so after the final B above, the next pitch class is C again) - any contiguous sequence of twelve pitch classes is know as an "octave". The convention when referring to a specific pitch is to subscript the pitch class with the octave position of the pitch in question on a standard grand piano. Thus F_3 refers to the F in the third octave (counting from the left) of a grand piano. Octaves begin on C and end on B. Middle C is thus notated C_4.

1 In this thesis we have used Narmour's convention of referring to melodic contour as registral direction.

Interval

The difference in pitch between two contiguous notes is called the interval, and is usually measured in semitones[2]. Intervals of certain sizes sound comfortable to our ears to greater and lesser extents. Figure B.3 lists the common names for intervals up to 14 semitones:

Interval size (in semitones)	**Interval Name (and abbreviation)**	**Note (for interval from C)**
0	Unison (u)	C
1	Minor Second (m2)	C# / Db
2	Major Second (M2)	D
3	Minor Third (m3)	D# / F#
4	Major Third (M3)	E
5	(perfect) Fourth (P4)	F
6	Diminished Fifth (A4/d5) (or Tritone or Augmented Fourth)	F# / Gb
7	(perfect) Fifth (P5)	G
8	Minor Sixth (m6)	G# / Ab
9	Major Sixth (M6)	A
10	Minor Seventh (m7)	A# / Bb
11	Major Seventh (M7)	B
12	Octave (P8)	C
13	Minor Ninth (m9)	C#/ Db
14	Major Ninth (M9)	D

Figure B.3 : Names of common intervals.

[2] A semitone is a single step in the well-termpered scale, such as a move up or down of one key (black or white) on a piano.

Registral direction (melodic contour)

Registral direction refers to the direction of pitch between two notes, i.e. it can be :

- ascent (second note has higher pitch than first),
- descent (second note has lower pitch than first),
- lateral (both notes have same pitch).

Time

Onset time

The point in time when a musical event begins (a note sounding or a rest beginning), is called the onset time.

Duration

The length of time of a musical event (for which a note plays or a rest lasts) is called the duration.

The context of melody

The final part of this section about melody outlines the two other aspects of music which form the context within which melody is heard: metre and harmony.

Metre

Metre refers to the hierarchy of regular pulses that can be heard in most Western Tonal Music. A measure of the metric strength of a note can be made, given the onset time of a given musical event.

Figure B.4 illustrates the metric hierarchy associated with $\frac{3}{4}$ time (i.e. bars of three crotchet beats, each of which can be subdivided into two half-beats). This view of a strictly hierarchical metre is useful for computational models, although there is some dispute about whether music is actually heard by listeners framed within such strict metre (the style of diagram for metric hierarchy is similar to those presented by Levitt (1985), Lerdahl and Jackendoff (1983) and Holland (1989)). Note that Figure B.4 illustrates the relative importance of notes on third- and sixth- bar boundaries, if appropriate one could continue dividing by two, to consider twelth- and twenty-fourth- bar boundaries. See Chapters 4 and 5 for details of how such relative metric importance is represented for M-PARSER.

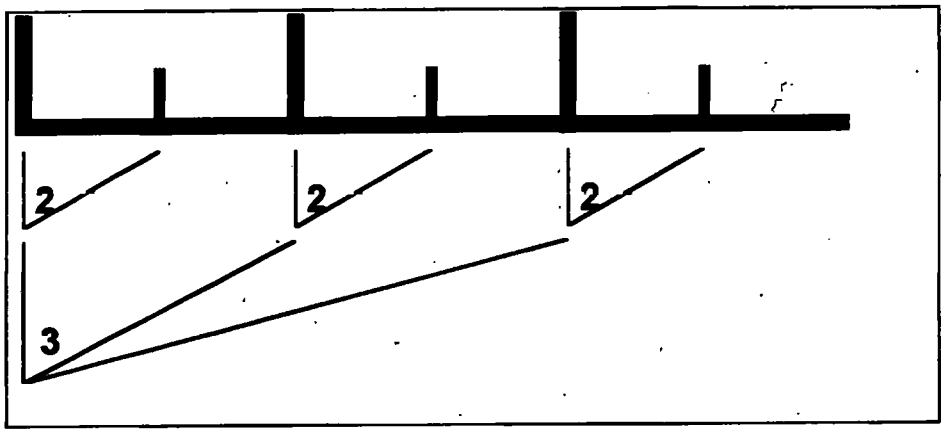

Figure B.4: Graphical representation of $\frac{3}{4}$ metre.

Harmony

Harmony refers to the effect of heard or implied notes in addition to those of a melody. Even when only a single line (one note at a time) melody is playing, underlying chords and scales are often implied, which a listener can infer — in most cases many alternative harmonisations are plausible.

Keys / modes / underlying scales

A melody exists in the context of (or can be thought of as implying) a subset of the twelve possible pitch classes, an example is the scale of C major comprising the notes C, D, E, F, G, A, B (the white notes on a piano keyboard). In addition to the WTM major and minor scales, other scales such as traditional (non-tonal) ***'modal'*** scales, can provide a useful harmonic perspective when analysing melody.

Chords

A chord is a collection of two or more notes, made up from a scale. Scales change very occasionally, if at all during a piece of music. However, while a melody plays the chord context may change comparatively frequently, perhaps once every two or four bars (sometimes more than once within a bar).

Appendix C: MOTIVE Time Units (MTUs) — A simple representation for time

A MOTIVE Time Unit (MTU) is a simple system of time representation for bars of music, allowing integer factorisation by common prime bar proportions. Every bar is 529200 MTUs long. This number is divisible by the primes 2, 3, 5 and 7, and calculated from the following equation:

$$1 \text{ bar} = 2^4 \times 3^3 \times 5^2 \times 7^2 = 529{,}200 \text{ MTUs}$$

Depending on time signature, specific note durations will vary in the number of MTUs they represent, with the respect to the proportion of the bar they occupy. For example in $\frac{4}{4}$ time, a crotchet has duration of a quarter of a bar, i.e. 529200 divided by 4 = 132300 MTUs, whereas in $\frac{3}{4}$ time a crotchet would have a duration of 176400 MTUs (529200 divided by 3).

Such a time representation allows melodies of varying time signatures and metres to be represented in a standard way. The representations used in M-PARSER were developed for simplicity — for more sophisticated systems general purpose music representations could be used (such as those proposed by Smaill et al (1994) and Balaban (1992)).

Appendix D:
The representation of melodies in MOTIVE

D.1. Three views of a melody

A melody is represented as the following four lists, plus an asserted predicate for each note of the melody:

Note_list a list of the unique identifiers for each note of the melody,

Metre_list a list of the metres associated with each part of the melody,

Scale_list a list of the harmonic scales associated with each part of the melody,

Chord_list a list of the chords associated with each part of the melody.

The lists can be thought of as reprepresenting three different views of the melody — a note-by-note view, a metric view, and an harmonic view (in terms of scales and chords).

D.2. The predicate for each note

Each note in the melody is represented by an asserted predicate, containing details of the following parameters.

Note_number A unique integer.

Pitch (the pitch of a note is stored in both a representation of the score symbol (***Pitch_class***, ***Octave***, and ***Accidental_list***) as well a the MIDI value for the pitch (***MIDI_pitch*** (an integer in the range 0..128, representing then number of semitones of the pitch above the C five octaves below middle C[1], or the atom **'rest'** if the melodic event is a rest).

Time_point An integer representing the onset time of the note, expressed in MTUs.

Duration An integer representing the duration of the note, expressed in MTUs.

Special Currently two things are stored here - prior calculations of beat strengths and harmonic closure strengnths. Beat strengths are rational numbers (e.g. $\frac{1}{2}$) representing the metric strength of each note from its onset time, given the time signature (see Smith (1993b) for more details). The harmonic closure strengths are currently entered by the user, but eventually will be calculated using a harmonic analysis tool.

D.3. The metre list

The metres associated with each part of a melody are represented as an element in a list of metres. Each element of the list is a pair as follows:

[***Onset_time***,]

The ***Onset_time*** is a valid note onset time (see Appendix C — MTUs).

The ***Metre*** element is a list as follows:

[[***Numerator, Demoninator***], ***Metric_vector*** , ***Note_value***]

The first element in the ***Metre*** list is a two-element list [***Numerator, Demoninator***] of the numerator and demoninator of the common music notation (CMN) of the metre (e.g. [4, 4] for 44, [3, 4] for 34, etc.).

---

1 The MIDI pitch value is the integer number of semitones of a pitch above C-1 (where middle C is C4). It should be notated that the harmonic role of the note (pitch class) is not represented — for example the MIDI pitch of 61 refers to the pitch between C4 and D4, but does not distinguish between the pitch classes of C# or Db (or B## etc.).

The second element of the ***Metre*** list is ***Metric_vector***[2]. This is a list describing how each bar for the metre can be subdivided. Each element of the ***Metric_vector*** list is either a simple integer or a list. Simple integers represent sudivisions of the bar — so a metric vector of [3, 2] means a bar can first be divided into 3, and then each third can be further divided into 2. If an element of a ***Metric_vector*** list is a list itself, it represents a bar sub-divided into an odd number of beats — the list element indicates the grouping of the beats. For example, a ***Metric_vector*** of [[3, 2], 2] represents a bar with 5 beats, grouped into 3 beats then 2; each of these 5 beats can then be further sub-divided into 2 sub-beats.

The third element of the ***Metre*** list is ***Note_value*** — the value of the note for each beat of the bar, e.g. quaver or semi-quaver.

---

[2] Metric Vectors are described in Smith (1990).

Appendix E:
The representation of analyses in MOTIVE

An analysis based on the Implication-Relisation Model takes the form of an ordered sequence of levels, each level containing an ordered sequence of structures. M-PARSER represents analyses as strict hierarchies (each level in the hierarchy is a sequence of structures made up of a subset of the notes in the level immidiately below). The structures for a level are contiguous (there are no notes for a given level that are not a member of a structure for that level).

Levels

In M-PARSER an analysis is represented by a list of ***level/5*** structures. Each such Prolog structure is as follows.

```
level(Level_num,List_of_struct_nums, Curr_struct,
   List_of_note_nums)
```

Level_num This is an integer, representing the number of the level (e.g. 2).

List_of_struct_nums This is a list of the IRM structure numbers that are associated with the level. The combination of the level number and the structure number (and if appropriate, chain letter) form a unique identifier for retrieving the IRM structure details from the structure list (see next sub-section). Due to the increased complexity when combinations and chains occur, the list is in the form of two types of elements:

s(N) where **'N'** is the number of a single structure.

ch(N,[Letter_list]) where **'N'** is the number of the combination or chain of structures, and **'Letter_list'** is an ordered list of letters, enumerating the structures in the combination or chain.

For example, if the first two structures for a level of an analysis were single structures, and then a combination of two structures occurred, the list of structure numbers would be: **[s(1), s(2), ch(3,[a,b])]**

Curr_struct This is a two element list, whose first element is the number of the current structure being parsed (for this level), and whose second element is either the atom **'new'** or the atom **'ongoing'** – indicating whether the structure has any notes stored on it yet (as soon as one structure is closed, the next one for the current level is begun, with the status as **'new'**).

List_of_note_nums This is a list of notes that have been promoted[1] to the current level, and have not yet been placed in a structure.

Structures

Each IRM structure represents a collection of notes which realise and deny the same implications. M-PARSER maintains a list of all structures, for all levels, when parsing a melody. There are two types of list element, differing only in an extra argument (a lower case letter) for structures that are part of a combination or chain — indicating the structure's position in the sequence of structures for the combination or chain. The two forms of the structure element are as follows.

```
structure(Level_num, Struct_num, Struct_class, Implications,
   Note_list)

structure(Level_num, Struct_num, Chain_letter, Struct_class,
   Implications, Note_list)
```

Level_num See previous section.

1 For the first analytical level (level one), this list contains all notes numbers of the melody (in sequence).

Struct_num This is an integer, representing the position of the structure for the level (i.e. the first structure is number one, the second number two, etc.) — the structures for a level are numbered in the order they are created during parsing, i.e. chronologically, from left to right.

[Chain_letter] (only when structure is part of a combination or chain) — This is a letter, indicating the position of the structure in the combination or chain (**'a'** for first, **'b'** for second, **'c'** for third etc.).

Struct_class This is an atom, indicating the type of structure (this variable is uninstantiated if there are less than three notes in the structure). The atom is a member of the following list: **[P, IP, VP, D, ID, R, IR, VR, (P), (IP), (VP), (D), (ID), (R), (IR), (VR)]**

Note_list This is a list of the notes that have been grouped as the structure (if the structure is still active, more note numbers may be added to the list) — the notes are in chronological sequence.

Appendix F:
Prolog encoding for primitives of the Implication-Realisation Model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               %%
%%       intervallic scale       %%
%%                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% small interval %%
intervallic_class(Interval,small) :-
    member(Interval,[0,1,2,3,4]). %%% [u,m2,M2,m3,M3]   always small

%% large interval %%
intervallic_class(Interval,large) :-
    Interval > 7. %%%%%%%% m6,M6,m7,M7,(P8) .... always large

%% threshold - so can be either %%
intervallic_class(Interval,small) :-
    member(Interval,[5,6,7]). %%%%%% [P4,tritone,P5] - threshold values
intervallic_class(Interval,large) :-
    member(Interval,[5,6,7]). %%%%%% [P4,tritone,P5] - threshold values

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               %%
%%         contour scale         %%
%%                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

reg_direction_scale([descent, lateral,ascent]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               %%
%%        duration scale         %%
%%                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

durational_scale([breve,semibreve,minim,crotchet,quaver,semiquaver,demisemiquav
er,hemidemisemiquaver]).
dotted_duration_scale([  [double_dotted_note,1.75],  [dotted_note,1.5],  [note,1]
]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%                                      %%
%%          MTU's for 4/4 time          %%
%%                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% for a 4/4 metre
numeric_duration([4,4],semibreve,529200).
numeric_duration([4,4],minim,264600).
numeric_duration([4,4],crotchet,132300).
numeric_duration([4,4],quaver,66150).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               %%
%%        beat strength scale    %%
%%                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

beat_strength_scale([1,1/2,1/3,1/4,1/5,1/6,1/7,1/8,1/9,1/16,-10]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                     %%
%%    interval between two notes            %%
%%                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
interval(Note1,Note2,
                        interval(Contour,Int_semitones,Int_class) ) :-
    note(Note1,P1,_,_,_,_,_),
    note(Note2,P2,_,_,_,_,_),
    reg_direction(P1,P2,Contour),
    semitones(P1,P2,Int_semitones),
    intervallic_class(Int_semitones,Int_class).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                          %%
%%          implications of an interval          %%
%%                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Imp_type : continuation / reveral
%% Int_class, Implied_int_class : large / small

implications( interval(Contour,_,Int_class),

implications(Imp_type,Implied_int_class,Implied_contour_list)) :-
    implication_type(Int_class,Imp_type),
   % implied note
    gestalt(Imp_type,registral_direction,Reg_gestalt),
    implied_reg_direction(Reg_gestalt,Contour,Implied_contour_list),

   % implied interval
    gestalt(Imp_type,interval,Int_gestalt),
    implied_intervallic_motion(Int_gestalt,Int_class,Implied_int_class).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%                                                                  %%
%%          gestalt implication from appropraite rule of inference
     %%
%%                                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

gestalt(continuation,similarity).
gestalt(reversal,differentiation).
```

Appendix G:
Extracts from MOTIVE Prolog procedure listings

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                              %%%
%%%          meta_parse                 %%%
%%%                              %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% meta_parse(+Curr_level,+Level_list,+Struct_list)
%
% Curr_level = number of level currently being parsed
% Level_list = list of level atoms
% Struct_list = list of struct atoms
%
% controlling routine to call "parse" and then "meta_action_list", to do
actions
% on levels and changing the Level and Struct lists
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% end of parse, so simply succeed
meta_parse(end_of_parse,_,_).

% call "parse" then call "meta_action_list" with the parse actions, then call
"meta_parse" recursively
meta_parse(Curr_level,Level_list,Struct_list) :-
	compress(['meta_parse, level: ',Curr_level],Big_word),
	big_heading(Big_word),
	parse(Curr_level,Level_list,Struct_list,Action_list),
	!,
	meta_action_list(Action_list,Curr_level,Level_list,Struct_list,New_level,
New_level_list,New_struct_list),
	!,
	meta_parse(New_level,New_level_list,New_struct_list).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                         %%%
%%%          meta_action_list         %%%
%%%                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% meta_action_list(+Action_list, +Curr_level, +Level_list, +Struct_list,
New_level, New_level_list, New_struct_list)
%
% Action_list - a list of actions to be performed for parsing
% Curr_level,Level_list,Struct_list = current details from meta_parse
% New_level,New_level_list,New_struct_list = new details, returned to
```

```
meta_parse
%
% call "meta_action" for each of the actions in "Action_list"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% parse is complete !!
meta_action_list([end_of_parse],Curr_level,Level_list,Struct_list,end_of_parse,
_,_).

% no more actions - meta_action_list is complete
meta_action_list([],Curr_level,Level_list,Struct_list,Curr_level,Level_list,Str
uct_list).

% do first action, then call "meta_action_list" again, with new level/struct
details
meta_action_list([Action|Rest_actions], Curr_level, Level_list, Struct_list,
Final_level, Final_Level_list, Final_struct_list) :-
      meta_action(Action,Curr_level, Level_list, Struct_list, New_level,
      New_level_list, New_struct_list),
      pp_status('after meta action : ', Action, New_level, New_level_list,
      New_struct_list),
      meta_action_list(Rest_actions,New_level,New_level_list,New_struct_list,
      Final_level, Final_Level_list,Final_struct_list).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                      %%%
%%%          meta_action                 %%%
%%%                                      %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% meta_action(+Action,+Curr_level,+Level_list,+Struct_list,-New_level,-
New_level_list,
% -New_struct_list)
%
% Action = an action to be performed for parsing
% Curr_level,Level_list,Struct_list = current details from meta_action_list
% New_level,New_level_list,New_struct_list = new details after action is
performed,
% returned to meta_action_list
%
% actions are :
%                                  ascend
%                                  descend
%                                  make_ongoing
%                                  mappend_note
%                                  promote(Note)
%                                  check_pros_closure
%                                  struct_implications(Imps)
%                                  retro_closure(Closure_cause)
%                                  close_curr_struct
%
% perform "Action" and return changes to level/struct details
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%% hierarchical level actions %%%%%%%%%

%%- ascend
%%%call parse at the next higher hierarchcial level
% a higher level than "Curr_level" exists, so make this the new current level
% (this should always be the case since at "top_level" the ascend action should
% not occur
meta_action(ascend,Curr_level,Level_list,Struct_list,New_level,Level_list,
Struct_list) :-
      poss_levels(Poss_level_list),
      three_part_list(_,Curr_level,[New_level|_],Poss_level_list).

%%- descend - 1
% error - cannot descend below level 1 !!
meta_action(descend,1,Level_list,Struct_list,New_level,Level_list,Struct_list)
:-
      !,
```

```
      matt_error("cannot descend below level 1 !!", meta_action,
      [descend,1,Level_list, Struct_list] ).

%%- descend - 2
%%%call parse at the next lower hierarchcial level
% a lower level than "Curr_level" exists, so make this the new current level
meta_action(descend,Curr_level,Level_list,Struct_list,New_level,Level_list,
Struct_list) :-
      poss_levels(Poss_level_list),
      three_part_list(Before,Curr_level,_,Poss_level_list),
      mappend(_,[New_level],Before).

%%- promote note - 1
% if next higer level is "top_level" don't bother with promotion
meta_action(promote(_),Curr_level,Level_list,Struct_list,Curr_level,Level_list,
Struct_list) :-
      poss_levels(Poss_level_list),
      reverse(Poss_level_list,[top_level,Curr_level|_]).

%%- promote note -
% don't promote "Note", since it has already been promoted
% (i.e.  is a member of the current level's "Special")
meta_action(promote(Note),Curr_level,Level_list,Struct_list,Curr_level,Level_li
st,
Struct_list) :-
      member( level(Curr_level,Special,_,_,_), Level_list ),
      member(Note,Special).

%%- promote note - 3
%%%mappend the indicated note to the note list for the next hiearchcial struct
% if there are at least two levels above the current one,
% and "Note" isn't a member of the current level's "Special" then
% mappend the indicated note to the next higher levels note list
% and mappend a note to the "Special" list on promoted notes for the current
level
meta_action(promote(Note),Curr_level,Level_list,Struct_list,Curr_level,New_leve
l_list2,
Struct_list) :-
      % check at least two levels above
      poss_levels(Poss_level_list),
      three_part_list(_,Curr_level,[Next_level,_|_],Poss_level_list),
      % check "Note" isn't a member of curr level's "Special"
      Old_level =
level(Curr_level,Special,Done_struct_nums,Curr_struct_details,
      Level_notes),
      three_part_list(Before,Old_level,Rest,Level_list),
      not( member(Note,Special) ),
      % append note to "Special" list for current level
      mappend(Special,[Note],New_special),
      New_level =
level(Curr_level,New_special,Done_struct_nums,Curr_struct_details,
      Level_notes),
      three_part_list(Before,New_level,Rest,New_level_list),
      % append "Note" to notes for next level
      Old_level_above = level(Next_level,Special2,Done_struct_nums2,
      Curr_struct_details2,Level_notes2),
      three_part_list(Before2,Old_level_above,Rest2,New_level_list),
      mappend(Level_notes2,[Note],New_level_notes2),
      New_level_above = level(Next_level,Special2,Done_struct_nums2,
      Curr_struct_details2,New_level_notes2),
      three_part_list(Before2,New_level_above,Rest2,New_level_list2).


%%%%%%%%%%%%%%% change structure details %%%%%%%%%%%%%%%%%%

%%- add note to structure
%%%mappend the indicated note to the indicated structure
% mappend "Note" to the note list for the current structure at level
"Curr_level"
meta_action(mappend_note,Curr_level,Level_list,Struct_list,Curr_level,New_level
_list,
```

```
New_struct_list) :-
        Old_level = level(Curr_level,Special,Done_structs,[Curr_struct,Status],
        [Next_note|Rest_level_notes]),
        three_part_list(Before_level, Old_level, After_level, Level_list),
        New_level = level(Curr_level,Special,Done_structs,[Curr_struct,Status],
        Rest_level_notes),
        three_part_list(Before_level, New_level, After_level, New_level_list),
        Old_struct = struct(Curr_level,Curr_struct,Struct_class,Implications,
        Struct_notes),
        three_part_list(Before,Old_struct,Rest,Struct_list),
        mappend(Struct_notes,[Next_note],New_struct_notes),
        New_struct = struct(Curr_level,Curr_struct,Struct_class,Implications,
        New_struct_notes),
        three_part_list(Before,New_struct,Rest,New_struct_list).

%%- close curr struct - 1
%%%move the indicated structure into the "done_structs" list for a level, and
%%%create a "new" structure
% move curr struct for curr level to "Done_structs" part of level entry
% get next structure number from "poss_structs" list
meta_action(close_curr_struct,Curr_level,Level_list,Struct_list,Curr_level,
New_level_list,Struct_list) :-
        Old_level = level(Curr_level,Special,Done_structs,[Curr_struct,ongoing],
        Level_notes),
        three_part_list(Before, Old_level, After, Level_list),
        mappend(Done_structs,[Curr_struct],New_done_structs),
        poss_structures(Poss_struct_list),
        three_part_list(_,Curr_struct,[New_curr_struct|Rest],Poss_struct_list),
        New_level =
level(Curr_level,Special,New_done_structs,[New_curr_struct,new],
        Level_notes),
        three_part_list(Before, New_level, After, New_level_list).

%%- close curr struct - 2
% if no more structures, error
meta_action(close_curr_struct,Curr_level,Level_list,Struct_list,_,_,_) :-
        poss_structures(Poss_struct_list),
        three_part_list(_,Curr_struct,[],Poss_struct_list),
        matt_error("run out of possible structures !!", meta_action,
        [close_curr_struct, Curr_level, Level_list, Struct_list]).


%%- make struct onging
%%%change the curr_struct from a "new" struct to "ongoing"
meta_action(make_ongoing,Curr_level,Level_list,Struct_list,Curr_level,New_level
_list,
Struct_list) :-
        Old_level = level(Curr_level,Special,Done_struct_nums,[Curr_struct,new],
        Level_notes),
        three_part_list(Before,Old_level,Rest,Level_list),
        New_level =
level(Curr_level,Special,Done_struct_nums,[Curr_struct,ongoing],
        Level_notes),
        three_part_list(Before,New_level,Rest,New_level_list).

%%- make struct retro
%%%????????

%%- test for prospective closure
%%%(self evident)
meta_action(check_pros_closure,Curr_level,Level_list,Struct_list,Curr_level,Lev
el_list,
Struct_list) :-
                %       metric_closure(Curr_level,Level_list,Struct_list,Metric),
                %
        durational_closure(Curr_level,Level_list,Struct_list,Durational),
        harmonic_closure(Curr_level,Level_list,Struct_list,Harmonic),
        combine_closures(Metric,Durational,Harmonic,Strenght),
        act_on_closure_strength(Strength).

act_on_closure_strength(_).
```

```
%%- test for promotion/merging
%%%(self evident)
meta_action(test_promotion_closure(Note,Closure_strengths),Curr_level,Level_lis
t,
Struct_list,Curr_level,New_level_list,Struct_list).
      %% to_be_written - currently a simply suceed %%

%% struct_impications(+Implications)
%%%% make "Implications" the current structures implications
meta_action(struct_implications(Implications),Curr_level,Level_list,Struct_list
,
Curr_level,Level_list,New_struct_list) :-
      Old_struct = struct(Curr_level,Curr_struct,Struct_class,_,Struct_notes),
      three_part_list(Before,Old_struct,Rest,Struct_list),
      New_struct = struct(Curr_level,Curr_struct,Struct_class,Implications,
      Struct_notes),
      three_part_list(Before,New_struct,Rest,New_struct_list).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                          %%%
%%%                parse                                                     %%%
%%%                                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parse(+Curr_level,+Level_list,+Struct_list,-Parse_actions)
%
% Curr_level = number of level currently being parsed
% Level_list = list of level atoms
% Struct_list = list of struct atoms
% Parse_actions = list of actions to perform for current situation
%
% check current state of notes for level, and structure, and check next note
% to decide action (e.g. append to curr structure, promote, ascend etc.)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 1 %
%next note is 'end of melody' + top level
parse(top_level,Level_list,_,[end_of_parse]) :-
      poss_levels(Poss_level_list),
      reverse(Poss_level_list,[top_level,Prev_level|_]),
      member(level(Prev_level,_,_,_,[end_of_melody]), Level_list).

% 2 %
%top level reached   (but not end of melody)
%      - descend
parse(top_level,Level_list,_,[descend]) :-
      poss_levels(Poss_level_list),
      reverse(Poss_level_list,[top_level,Prev_level|_]),
      not( member(level(Prev_level,_,_,_,[end_of_melody]), Level_list) ).

% 3 %
%next note is 'end of melody' + not top level
%      - close curr struct
%      - promote last note
% - promote "end_of_melody"
%      - ascend
parse(Curr_level,Level_list,Struct_list,[close_curr_struct,promote(Last_note),
promote(end_of_melody),ascend]) :-
      not( Curr_level = top_level),
      member( level(Curr_level,_,_,[Curr_struct,ongoing],[end_of_melody]),
Level_list),
      member( struct(Curr_level,Curr_struct,_,_,Struct_notes), Struct_list),
      reverse(Struct_notes,[Last_note|_]).

% 4 %
%no-notes for current level
%      - descend
parse(Curr_level,Level_list,Struct_list,[descend]) :-
      member(level(Curr_level,_,_,_,[]), Level_list).
```

```
% 5 %
%adding first note for an new structure
%       - promote note (always ??)
%       - ascend
parse(Curr_level,Level_list,Struct_list,[make_ongoing,mappend_note,check_pros_c
losure,
promote(Next_note)]) :-
        member(level(Curr_level,_,_,[Curr_struct,new],[Next_note|_]),
Level_list).

% 6 %
%add second note to structure - and calc implications
parse(Curr_level,Level_list,Struct_list,[struct_implications(Imps),mappend_note
,
check_pros_closure]) :-
        member( level(Curr_level,_,_,[Curr_struct,ongoing],[Next_note|_]),
Level_list),
        member( struct(Curr_level,Curr_struct,_,_,[Struct_note]), Struct_list),
        !, %% now we know only one note in struct so far
        interval(Struct_note,Next_note,Interval),
        implications(Interval,Imps).

% 7 %
%add second note to structure - and calc implications
parse(Curr_level,Level_list,Struct_list,[struct_details(Struct_name,Struct_attr
ibutes),
mappend_note, check_pros_closure]) :-
        member( level(Curr_level,_,_,[Curr_struct,ongoing],[Next_note|_]),
Level_list),
        member(
struct(Curr_level,Curr_struct,_,Imps,[Struct_note1,Struct_note2]),
        Struct_list),
        !, %% now we know only two notes in struct so far
        interval(Struct_note2,Next_note,Interval),
        struct_class(Imps,Interval,Struct_name,Struct_attributes).

% 8 %
%test new note for retro closure
parse(Curr_level,Level_list,Struct_list,[retro_closure(Closure_cause)]) :-
        member( level(Curr_level,_,_,[Curr_struct,ongoing],[Next_note|_]),
Level_list),
        member( struct(Curr_level,Curr_struct,_,Imps,Struct_notes), Struct_list),
        reverse(Struct_notes,[Prev_note|_]),
        retro_closure(Next_note,Struct_notes,Prev_note,Implications,Closure_cause
).

% 9 %
%new note meets implications for curr structure
%       - add note to structure
%       - test for prospective closure
parse(Curr_level,Level_list,Struct_list,[mappend_note,check_pros_closure]).

% 10 %
%new note meets struct, if struct is change to retro
%       - add note to struct
%       - make struct retro

%%------------ meta_actions to deal with retro_closure --------

%retro closure when testing next note (e.g. stopping or rest)
%       - close structure
%       - make new structure
%       - promote last note of prev struct
%       - ascend
%parse(Curr_level,Level_list,Struct_list,[close_curr_struct,
promote(Prev_note), ascend]) :-
        Struct_pred = struct(Curr_level, Curr_struct,_,_,
[Curr_note|Struct_notes]),
        member(Struct_pred,Struct_list),
        Level_pred =
level(Level_num,Special,Prev_structs,Curr_struct,Level_notes),
```

```
member(Level_pred,Level_list),
retrospective_closure(Curr_note,Level_pred,Struct_pred),
reverse(Struct_notes,[Prev_note]).
```

Appendix H:
Feasibility of each of MELODY-EDs components

In this appendix we consider the feasibility of implementing each of the components identified as required to support such scenarios as would occur when students use the MELODY-ED outline architecture described in Chpater 8.

H. 1. The TOOLBOX

MOTIVE, the constraint-based tool described in the next section, is an appropriate tool to be included in the "toolbox" component of MELODY-ED. We have identified a need for a tool (or set of tools) that can manipulate melodies in terms of constraints, and MOTIVE is a tool that allows both the analysis and generation of melodies using a single formalism — an *Implication-Realisation Model* analysis (using optional, additional constraints during generation).

It is intended that MOTIVE be an interactive, audio-graphical tool (see Chapter 8), providing real time analysis and generation of melodies. The tool represents a medium in which student, tutor and system can communicate in terms of constraints and fragments of melodies, allowing representation of notes, metre, harmony and other constraints in a principled way.

H.2. Direct manipulation

In Chapter 8 we have presented a design for a direct manipulation interface for MOTIVE.

H.3. The corpus of musical pieces

This component is straightforward to implement as a simple database of melodies, for example using the melody representation presented in Chapter 5.

H.4. The constraint engine

Existing AI work has for some time led to the development of constraint-satisfaction engines (for example, Winston, 1981). In Chapter 7 we have presented a discussion on how the constraint-satisfaction process has been implemented in the current prototype of MOTIVE, and highlighted a number of ways such constraint-satisfaction could be made more efficient.

H.5. A musical constraint knowledge base

A musical constraint knowledge base could be compiled, including the kinds of constraints discussed in the Chapter 7 (for metre, harmony and phrase-variation). Many aspects of music theory in general have been modelled as constraints in previous work (e.g. Holland 1989, Levitt 1985).

H.6. A tutor

A musically knowledgable human tutor is currently envisaged to play the role of guiding the students use of the MOTIVE tool. However, as we tentatively discuss in part of Chapter 8, it may be possible to use an AI tutor to play this role (this is subject to considerable further work investigating the possibility).

References

Allen (1987) J. Allen. Natural Language Understanding. Benjamin/Cummings, Menlo Park, CA, USA.

Baker (1989a) Michael J. Baker. "An artificial intelligence approach to musical grouping analysis". *Contemporary Music Review*, 3: 43-68, Harwood Acdemic Publishers, GmdH.

Baker (1989b) Michael J. Baker. "An computational approach to modeling musical grouping structures". *Contemporary Music Review*, 4: 311-325, Harwood Acdemic Publishers, GmdH.

Baker (1989c) Michael J. Baker. *Negotiated Tutoring: An Approach to Interaction in Intelligent Tutoring Systems*. PhD thesis, Centre for Information Technology in Education, The Open University, UK.

Baker (1992) Michael J. Baker. "Design of an Intelligent Tutoring System for Musical Structure and Interpretation". In: *Understanding Music with AI: Perspectives on Music Cognition*, Mira Balaban, Kemal Ebcioglu and Otto Laske (eds.), AAAI Press/MIT Press, Cambridge, MA, USA.

Baker (1995) Michael J. Baker. Personal communication, Milton Keynes, UK, November 1995.

Balaban (1992) Mira Balaban. "Music Structures: Interleaving the Temporal and Hierarchical Aspects in Music". In: *Understanding Music with AI: Perspectives on Music Cognition*, Mira Balaban, Kemal Ebcioglu and Otto Laske (eds.), AAAI Press/MIT Press, Cambridge, MA, USA.

Balzano (1982) G. J. Balzano. "The pitch set as a level of description for studying musical pitch perception". In *Music, Mind and Brain: the neuropsychology of music*, M. Clynes (Ed.), Plenum, New York, USA.

Bamberger (1977) Jean Bamberger. "In search of a tune". In *The arts and cognition* (pp. 284-319), Johns Hopkins Univeristy, Balimore, USA.

Baroni & Jacoboni (1978) Mario Baroni and Carlo Jacoboni, "Proposal for a Grammar of Melody". Univerita di Bologna, Les Presses de L'Universite de Monreal.

Bharucha (1994) Jamshed J. Bharucha. "Tonality and Expectation". In *Musical Perceptions*, Rita Aiello (Ed.), Oxford University Press, Oxford, UK.

Brady & Berwick (1983) Michael J. Brady and Robert C. Berwick (Eds.). *Computational Models of Discourse*, MIT Press, Cambridge, MA, USA.

Bower & Hildgard (1971) Gordon H. Bower and Ernest R. Hildgard. *Theories of Learning*. Prentice-Hall, New Jersey, USA.

Bull & Smith (1995) Susan Bull and Matt Smith. Using Targeted Negotiation to Support Students Learning. *In Procededings of the International Conference for Computers in Education 1995*, Singapore.

Burns & Ward (1982) Edward M Burns and W. Dixon Ward. "Intervals, scales and tuning". In *The psychology of music*, Diana Deutsch (Ed.), Academic Press, New York, USA.

Butler (1992) David Butler. "Reviews: Eugene Narmour, The Analysis and Cognition of Basic Melodic Structures". *Music Perception*, 10 (2): 243-251.

Calsen (1981) J. C. Carlsen. "Some factors which influence melodic expectency". *Psychomusicology*, 1: 12-29.

Cerri et al. (1986) S. A. Cerri, P. Landini, and M. Leoncini. "Cooperative agents for knowledge-based information systems". In: *Applied Artificial Intelligence Journal* 1 (1).

Chomsky (1957) Noam Chomsky. *Syntactic Structures*. Mouton, The Hague.

Clarke (1986) Eric Clarke. "Theory, analysis and the psychology of music: A critical evaluation of Lerdahl, F.and Jackendoff, R. A Generative Theory of Tonal Music". *Psychology of Music*, 14: 3-16.

Coates (1994) Daran Coates. "Representations of the MONK harmonisation systems". In *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Colley et al (1992) A. Colley, L. Banton, J. Down and A. Pither. "An expert-novice comparison in musical composition". *Psychology of Music*, 20 (1): 124-137.

Colmerauer (1978) A. Colmerauer. "Metamorphosis grammars". In (pp. 133-189) *Natural Language Communication with Computers*, Lecture Notes in Computer Science, L. Bolc (Ed.), Springer-Verlag, New York, USA.

Cook (1994) John Cook. "Agent Reflection in an Intelligent Learning Environment Architecture for Musical Composition". In *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Cook & Morgan (1995) John Cook and Nigel Morgan. "COLERIDGE: Composition Learning Environment for Reflection about Intentions and Dialogue Goals in Education", in *Proceedings of the International Congress In Music and Artificial Intelligence*, Edinburgh, UK.

Cumming (1992) Naomi Cumming. "Eugene Narmour's Theory of Melody". *Music Analysis*, 11 (203): 354-374.

Davidson & Welsh (1988) L. Davidson and P. Welsh. "From collections to structure: The developmental path of tonal thinking". In *Generative processes in music: The psychology of performance, improvisation, and composition* (pp. 260-285), John A. Sloboda (Ed.), Clarendon Press, Oxford, UK.

Dechter (1990) R. Dechter. "Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition". *Artificial Intelligence*, 41: 273-312.

Dechter and Meiri (1989) R. Dechter and I Meiri. "Experimental evaluation of pre-processing techniques in constraint satisfaction problems". In: *Proceedings of IJCAI-89*, pp. 271 - 277, held Detroit, Michigan, USA.

Dechter and Pearl (1987) R. Dechter and J. Pearl. "Network-based heuristics for constraint-satisfacton problems". *Artificial Intelligence*, 34: 1-38.

De Furia & Scacciaferro (1988) Steve De Furia and Joe Scacciaferro. *The MIDI Book: Using MIDI and Related Interfaces*. Third Earth Productions, Inc., Pompton Lakes, NJ, USA.

Desain & Honning (1986) Peter Desain and Henkyon Honing. "LOCO: Composition microworlds in logo". In *Proceedings of the International Computer Music Conference*, The Hague, Netherlands.

Deutsch (1982a) D. Deutsch. *The Cognitive Psychology of Music*. Academic Press, New York, USA.

Deutsch (1982b) D. Deutsch. "The processing of pitch combinations". In D. Deutsch (Ed.): *The psychology of music*, Academic Press, New York, USA.

Deutsch & Boulanger (1984) D. Deutsch & R. Boulanger. "Octave equivalence and the immediate recall of pitch sequences". *Music Perception*, 2: 40-51.

Deutsch & Feroe (1984) D. Deutsch & J. Feroe. "The internal representation of pitch sequences in tonal music". *Psychologal Review*, 88: 503-522.

Dowling (1986) W. J. Dowling. *Music Cognition*. Acdemic Press, New York, USA.

Dowling & Harwood (1986) W. J. Dowling & D. L. Harwood. *Music Cognition*. Academic Press, Orlando, USA.

Ebcioglu (1986) Kemal Ebcioglu. *An Expert System for Harmonization of Chorales in the Style of J. S. Bach*. PhD thesis, Technical Report TR 86-09, Department of Computer Science, S.U.N.Y. at Buffalo, USA.

Ebcioglu (1992) Kemal Ebcioglu. "An Expert System for Harmonizing Chorales in the style of J. S. Bach". In *Understanding Music with AI: Perspectives on Music Cognition*, Mira Balaban, Kemal Ebcioglu and Otto Laske (Eds.), AAAI Press/MIT Press, Cambridge, MA, USA.

Edelman (1987) Gerald M. Edelman. *Neural Dawinism: The Theory of Neuronal Group Selection*. Basic Books, New York, USA.

Elsom-Cook (1989) Mark Elsom-Cook (Ed.). *"Guided Discovery Learning Systems"* . Chapman and Hall, London.

Fenton (1992)A. Fenton. "The design of an intelligence tutoring system for music". *Musicus*, 1(ii): 125-143.

Fodor (1983) Jerry A. Fodor. *The Modularity of Mind: An Essay on Faculty Psychology*. MIT Press, Cambridge, MA, USA.

Fraisse (1982)P. Fraisse. *Psychologie du rhythme*. Publications Universitaires de Louvain,.

Freuder (1978) E. C. Freuder. Synthesizing constrain expressions. *Communications of the ACM*, 21: 958-966.

Gaschnig (1979) J. Gaschnig.*Performance measurement and analysis of certain search algorithms*. Technical Report CMU-CS-79-124, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Greenberg (1986) Gary Greenberg. "Computers and Music Education: A Compositional Approach". In *Proceedings of the International Computer Music Conference*.

Griffith (1994) Niall Griffith. Personal communication. Department of Computer Science, Exeter University, UK.

Gross (1984) D. Gross. "Computer Applications to Music Theory: a retrospective". *Computer Music Journal*, 8 (4).

Guesgen and Hertzberg (1991) Hans Werner Guesgen and Joachim Hertzberg. *Advanced Constraint Techniques: An AISB Tutorial*. German National Research Centre for Computer Science (GMD), Expert Systems Research Group, Schloss Birlinghoven, Germany.

Haralick and Elliot (1980) R. M. Haralick and G. L. Elliot. "Increasing tree search efficiency for constraint satisfaction problems". *Artificial Intelligence*. 14: 263 - 313.

Heidorn (1975) G. E. Heidorn. "Augmented phrase structure grammar". In *Theorectical Issues in Natural Language*, Schank and Nash-Webber (Eds.), Association for Computational Linguistics.

Holland (1989) Simon Holland. *Artificial Inteliegence, Education and Music*. Unpublished PhD thesis, IET, Open University, UK.

Holland and Elsom-Cook (1990) Simon Holland and Mark Elsom-Cook. "Architecture of a Knowledge-based Music Tutor". In Mark Elsom-Cook (Ed.) *Guided Discovery Learning Systems*. Chapman and Hall, London.

Holland (1991) Simon Holland. *Preliminary report on the design of a constraint-based musical planner*. Research report AUCS/TR9113, Department of Computing Science, Aberdeen University, UK.

Holland (1994) Simon Holland. "Learning About Harmony with Harmony Space: An Overview". In *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Howell et al (1985) P. Howell, Ian Cross & R. West (eds.) *Musical Structure and Cognition*. Academic Press, London, UK.

Johnson-Laird (1988) Philip Johnson-Laird. *The Computer and The Mind (An Introduction to Cognitive Science)*. Fontana Paperbacks, London. UK.

Kaipainen et al. (1995) Mauri Kaipainen, Petri Toiviainen, and Jukka Louhivuori. "A Self-Organising Map that Generates Melodies: Modelling and Testing Spontaneously Developing Implicit Grmmars", in *Proceedings of the International Congress In Music and Artificial Intelligence*, Edinburgh, UK.

Katz (1951) David Katz. *Gestalt Psychology: Its Nature and Significance* (Translated by Robert Tyson). Methuen and Co. Ltd., London, UK.

Koffka (1922) Kurt Koffka. "Perception: An introduction to the Gestalt-Theorie". *Psychological Bulletin*, 19 (10).

Koffka (1935) Kurt Koffka. *Principals of Gestalt Psychology*. Routledge and Kegan Paul Ltd., London, UK.

Kratus (1985) John Kratus. *Rhythm, Melody, Motive, and Phrase Characteristics of Original Songs by Children Aged Five to Thirteen*. Unpublished PhD thesis, Northwstern University, USA.

Kratus (1991) John Kratus. "Characterization of the compositional strategies used by children to compose a melody". *Canadian Journal of Research in Music Education*, 33: 95-103. (Special ISME Research Edition).

Krebs (1989) Harald Krebs. Review of "Chopin Studies" (Samson (Ed.), 1988). In the *Journal of Music Theory*, 33 (2): 429-439.

Krumhansl (1979) C. Krumhansl. "The psychological representation of musical pitch in a tonal context". *Cognitive Psychology*, 11: 346-374.

Krumhansl (1983) C. Krumhansl. "Perceptual structures for tonal music". *Music Perception*, 1(1): 28-62.

Krumhansl (1991) C. Krumhansl. "Melodic Structure: Theoretical and perceptual aspects". In J. Sundberg, L. Nord & R. Carlson (eds.), *Music, language, speech and brain*. Macmillan, London, UK.

Krumhansl, Bharucha & Kessler (1982) C. Krumhansl, J. J. Bharucha & E. Kessler. "Perceived harmonic structure of chords in three related musical keys". *Journal of Experimental Psychology: Human Perception & Performance*, 8: 24-36.

Krumhansl & Kessler (1982) C. Krumhansl & E. Kessler. "Tracing the dynamic changes in perceioved tonal organiszation in a spatial representationof musical keys". *Psychological Review*, 89: 334-368.

Krumhansl & Schellenberg (1990) C. Krumhansl & E. Schellenberg. "An empirical test of the Implication-Realisation model". Paper presented at the *Conference on Music and the Cognitive Sciences*, 17-21 September, Cambridge, UK.

Krumhansl & Shepard (1979) C. Krumhansl & R. Shepard. "Quantification of the hierarchy of tonal functions within a diatonic context". *Journal of Experimental Pschyoloy: Human Perception and Performance*, 5: 579-594.

Laske (1980) Otto Laske. "Towards an explicit Cognitive Theory of Musical Listening". *Computer Music Journal*, 4 (2): 73-83.

Laske (1988) Otto Laske. "Introduction to Cognitive Musicology". *Computer Music Journal*, 12 (1): 43-56.

Lee (1991) Christopher S. Lee. "The Perception of Metrical Structure: Experimental Evidence and a Model". In *Representing Musical Structure*, Howell, West and Cross (eds.), Academic Press Ltd.

Leman (1989) Marc Leman. "Symbolic and Subsymbolic Information Processing in Models of Musical Communication and Cognition". *Interface*, 18: 141-160.

Leman (1992) Marc Leman. "Artificial Neural Networks in Music Research". In *Computer Representations and Models in Music*, Alan Marsden and Anthony Pople (Eds.), Academic Press Ltd, UK.

Lerdahl (1988) Fred Lerdahl. Tonal Pitch Space. *Music Perception*, 5 (3): 351-350.

Lerdahl & Jackendoff (1983) Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, MA, USA.

Lester (1979) J. Lester. "Articulation of tonal structures as a criterion for analytic choices". *Music Theory Spectrum*, I: 69-79.

Levitt (1981) David A. Levitt. *A Melody Description System for Jazz Improvisation*. Unpublished Masters thesis, Department of Electrical Engineering and Computer Science, MIT, USA.

Levitt (1984) David A. Levitt. "Constraint Languages". *Computer Music Journal*, 8 (1).

Levitt (1985) David A. Levitt. *A Representation for Musical Dialects*. Unpublished PhD thesis, Department of Electrical Engineering and Computer Science, MIT, USA.

Longuet-Higgins (1962) H. Christopher Longuet-Higgins. "Two letters to a musical friend". In: *The Music Review*, November 1962, 23: 244-228 & 271-280. (Reprinted in H. Longuet-Higgins, *Mental processes: Studies in cognitive science*. The MIT Press, Cambridge, MA, USA, 1987)

Longuet-Higgins & Lee (1984) H. Christopher Longuet-Higgins and Christopher Lee. "The rhythmic interpretation of monophonic music". *Music Perception*, 1: 424-441.

Longuet-Higgins & Lee (1982) H. Christopher Longuet-Higgins and Christopher Lee. "The perception of musical rhythm". *Perception*, 11: 115-128.

Loy (1985) G. Loy. "Musicians make a standard: The MIDI phenomenon". *Computer Music Journal*, 9 (4): 8-26.

Markus (1980) Mitch Markus. *Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA, USA.

Marr (1982) David Marr. *Vision*. Freeman, San Francisco, USA.

Massero (1972) D. W. Massero. "Perceptual images, processing time and perceptual units in auditory perception". *Psychological Review*, 79: 124-145.

McDonald (1983) David D. McDonald. "Natrural Language Generation as a Computational Problem". In*Computational Models of Discourse*, Michael J. Brady and Robert C. Berwick (Eds.), MIT Press, Cambridge, MA, USA.

Minsky (1980) Marvin Minsky (interviewed by Curtis Roads). *Interview with Marvin Minsky*. Computer Music Journal, 4 (3), pp.25-39.

Meyer (1956) Leonard B. Meyer. *Emotion and Meaning in Music*. University of Chicago Press, Chicago, USA.

Mursell (1937) J. L. Mursell. *The Psychology of music*. Norton, New York, USA.

Narmour (1984) Eugene Narmour. "Some Major Theoretical Problems Concerning the Concept of Hierarchy in the Analysis of Tonal Music". *Music Perception*, 1 (2): 129-199.

Narmour (1989) Eugene Narmour. "The 'genetic code' of melody: Cognitive structures generated by the implication-realization model". *Contemporary Music Review*, 4: 45-63.

Narmour (1990) Eugene Narmour. *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. University of Chicago Press, Chicago, USA.

Narmour (1991) Eugene Narmour. "The Top-Down and Bottom-Up Systems of Musical Implication: Building on Meyer's Theory of Emotional Syntax". *Music Perception*, 9 (1): 1-26.

Narmour (1992) Eugene Narmour. *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*, University of Chicago Press, Chicago, USA.

Oppenheim (1993) Danny V.Oppenheim. "Dmix: A multi-faceted environment for composing and performing computer music: its design, philosophy, and implementation". In *Proceedings of the SEAMUS Conference*, Austin, Texas, USA. (Also in proceedings of the Arts and Technology Symposium, Conecicut College, Connecticut, USA).

Oppenheim (1994) Danny V.Oppenheim. "Slappability: A New Metaphor for Human Coputer Interaction". In *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Ovans (1992) Russell Ovans. *Efficient Music Composition via COnsistency Techniques*. Simon Fraser Univserity, Centre for Systems Science, Technical Report CSS-IS TR 92-02, Burnaby, B. C., Canada.

Peel & Slawson (1984) J. Peel and W. Slawson. "Review of A Generative Theory of Tonal Music by F. Lerdahl & R. Jackendoff". *Jounral of Music Theory*, 28: 271-294.

Perkins (1981) David N. Perkins. *The Mind's Best Work*. Harvard University Press, Cambridge, MA, USA.

Pollock (1989) John L. Pollock. *How to Build a Person: a Proloegomenon*. MIT Press, Cambridge, MA, USA.

Pomerantz (1981) James R. Pomerantz. "Perceptual organization in Information Processing". In *Perceptual Organization*, Michael Kubovy and James R. Pomerantz (Eds.). Erlbaum, New Jersey, USA.

Reimer (1989) B. Reimer. *A Philosophy of Music Education.* Prentice Hall, New Jersey, USA.

Roads (1985) Curtis Roads. *Research in Music and Artificial Intelligence.* ACM Computing Srveys, 17 (2), pp.163-190.

Rosner (1984) Burton S. Rosner. "Review of A generative theory of tonal music, by Fred Lerdahl and Ray Jackendoff". *Music Perception*, 2: 275-290.

Rosner (1988) Burton s. Rosner. "Music perception, music theory, and psychology". In *Explorations in music, the arts and ideas*, Eugene Narmour and Ruth Solie (eds.), Stuyvestant, Pendgragon, New York, USA.

Rosner & Narmour (1992) , Burton S. Rosner and Eugene Narmour. "Harmonic Closure: Music Theory and Perception". *Music Perception*, 9 (4): 383-411.

Rowe (1993) Robert Rowe. "Music Theory, Music Cognition" (chapter 4). In *Interactive Music Systems - machine listening and composing.* MIT Press, Cambridge, MA, USA.

Samson (1988) Jim Samson (Ed.). *Chopin Studies.* Cambridge Univerrsity Press, Cambridge, UK.

Schellenberg & Krumhansl (1991) E. Schellenberg and C. Krumhansl. *Testing the implication-realisation model cross-culturally.* Paper presented at the annual meeting of the Americal Psychological Society, 13-16 June, Washington DC, USA.

Schenker (1979) Heinrich Schenker. *Free Composition.* Longmans. English Translation of "Der Freie Satz" (originally 1935, translated and edited by Ernst Oster).

Schoenberg (1911 / 1978) Arnold Schoenberg. *Theory of Harmony.* Originally published 1911. Translated by R. Carter, University of California Press, Berkley, CA, USA, 1978.

Schoenberg (1943) Arnold Schoenberg. *Models for Beginners in Composition.* Schirmer, New York, USA.

Schoenberg (1967) Arnold Schoenberg. *Fundamentals of music composition.* Faber and Faber, London, UK.

Schwanauer & Levitt (1993) S. M. Schwanauer and David A. Levitt (eds.). *Machine Models of Music.* MIT Press, Cambridge, MA, USA.

Sharples & O'Malley (1987) Mike Sharples and Claire E. O'Malley. "A Frameowrk for the Design of a Writer's Assistant". In *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction,* John Self (Ed.), Chapman and Hall, London, UK.

Shepard (1982) R. N. Shepard. *Mental images and their transformations.* The MIT Press, Cambridge, MA, USA.

Shneiderman (1982) Ben Shneiderman. "The future of interactive systems and the emergence of direct manipulation". *Behaviour and Informatio Technology,* 1: 237-256.

Slack (1984) Jon M. Slack. Cognitive Science Research. In *Artificial Intelligence: Tools, Techniques and Applications,* Tim O'Shea and Marc Eisenstadt (Eds.), Harper and Row, New York, USA.

Sleeman (1983) Derek H.Sleeman. "Inferring student models for intelligent computer-aided instruction". In *Machine Learning: An Artificial Intelligence Aproach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.), Morgan kaufmann, Palo Alto, California, USA.

Sloboda (1985) John Sloboda. *The Musical Mind: The cognitive psychology of music*. Oxford University Press, New York, USA

Sloboda (1988) John Sloboda (ed.). *Generative Processes in Music: The Psychology of Performance, Improvisation and Composition*. Oxford University Press, New York, USA.

Smaill et al (1994) Alan Smaill, Geraint A. Wiggins and Eduardo Miranda. "Music Representation — Between the Musician and the Computer". In: *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Smith (1990) Matt Smith. *Modelling of Melody Generation and Musical Dialects*. Unpublished Masters thesis, Department of Computing Science, University of Aberdeen, UK.

Smith (1991) Matt Smith. *Proposals for computational models of melody*. Department of Computing Science Technical Report AUCS/TR9117, University of Aberdeen. UK.

Smith (1992) Matt Smith. An AI tool for melody analysis and generation. In: *Artificial Intelligence and Music* (notes for the workshop held at ECAI-92, Vienna, Austria, August 3-7 1992).

Smith (1995) Matt Smith. "CONNIE: An intelligent learning environment for creative tasks based on the negotiation of constraints". In *Proceedings of the World Conference on Artificial Intelligence in Education 1995*, Washington, USA.

Smith & Holland (1992a) Matt Smith and Simon Holland. "An Intelligent Tutor for Melody Composition". In: *Advances in Artificial Intelligence - Theory and Application* (Volume II of the Procedings of the 6th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 17- 23 1992), John W. Brahan and George E. Lasker (eds.). International Institue for Advanced Studies in Systems Research and Cybernetics.

Smith & Holland (1992b) Matt Smith and Simon Holland. "A computer tool for melody analysis and generation". In: *Proceedings of the International Computer Music Conference*, San Jose, USA.

Smith & Holland (1994) Matt Smith and Simon Holland. "MOTIVE: The Development of an AI Tool for Beginning Melody Composers". In: *Music Education: An Artificial Intelligence Approach*, Matt Smith, Alan Smaill and Geraint A. Wiggins (eds.), Springer-Verlag, Berlin, Germany.

Smoliar (1980) Stephen W. Smoliar. "A computer aid for Schenkerian analysis". *Computer Music Journal*, 4 (2).

Smoliar (1991) Stephen W. Smoliar. "Review: The Analysis and Cognition of Basic Melodic Structures: The Implication-Realisation Model by Eugene Narmour". *In Theory Only*, 12 (1-2): 43-56.

Sowa (1984). John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA, USA.

Stallman and Sussman (1977) R. M. Stallman and G. J. Sussman. "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis". *Artificial Intelligence*, 9: 135-196.

Steedman (1983) Mark J. Steedman. "A Generative Grammar for Jazz". *Music Perception*, 2 (1).

Sundberg & Lindblom (1976) J. Sundberg and B Lindblom. Generative theories in language and music descriptions. *Cognition*, 4: 99-122.

Tenney & Polasnki (1980) James Tenney and Larry Polanski. "Temporal gestalt perception in music". *Journal of Music Theory*, 24: 205-41.

Thomas (1985) Marilyn Taft Thomas. "Vivace: A rule-based AI system for composition". In *Proceedings of the International Computer Music Conference*, Vancouver, Canada.

Todd (1989) Peter M. Todd. "A Connectionist Approach to Algorithmic Composition". *Computer Music Jounral*, 13 (4): 27-43.

Ulrich (1977) John W. Ulrich. "The Analysis and Synthesis of Jazz by Computer". In: *Proceedings of the Fifth IJCAI, Vol. 2*, Department of Computer Science, Carnegie-Mellon University, CA, USA.

Waltz (1972) D. L. Waltz. *Generating semantic descriptions from drawings of scenes with shadows*. Technical Report AI-TR-271, Massachusetts Institute of Technology, Cambridge, Massachusets, USA.

Warburton (1960) Annie O. Warburton. *Melody Writing and Analysis*, (second edition), Longman Group UK Ltd.

Walker *et. al* (1987) Adrian Walker (Ed.). Michael McCord, John F. Sowa, and Walter G. Wilson (Authors). *Knowledge Systems and Prolog: A Logical Approach to Expert Systems and Natural Language Processing*. Addison-Weslesy Publishing Company, Inc. Reading, MA, USA.

Weber (1817) G. Weber. *Theory of Musical Composition*. Cocks, London, UK. (Translated 1851)

Weber (1824) G. Weber. *Versuch einer Geordeneten Theorie*. Mainz: B. Schotts Sohne.

Webster (1989) P. Webster. "Composition software and issues surrounding its use in research settings with children". *Psychomusicology*, 8(2): 163-169.

Wenger (1987) E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers Inc., Los Altos, Califronia, USA.

White (1981) Barbara White. *Designing Computer Games to Facilitate Learning*, PhD thesis. Technical Report AT-TR-619, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Winograd (1983) T. Winograd. *Language as a Cognitive Process, Volume 1: Syntax*. Addison-Wesley.

Winsor (1987) P. Winsor. *Computer assisted music composition*. Petrocelli Books, Princetown.

Winston (1981) Partick Henry Winston. *Artificial Intelligence*. Addison-Wesley, New York, USA.

Woodrow (1911) H. A. Woodrow. "The role of pitch in Rhythm". *Archives of Psychology*, 14: 1-66.

Wuorinen (1979). Charles Wuorinen. *Simple Composition*. Longman, New York, USA.

Xenakis (1985) I. Xenakis. "Music Composition Treks". In *Composers and the Computer*, Curtis Roads (Ed.), Kaufmann, Los Altos, USA.

Young & O'Shea (1981) Richard Young and Tim O'Shea. "Errors in children's subtraction". *Cognitive Science*, 5: 153 - 177. (Originally in *Proceedings of the Meeting of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, 1978).